Introduction

Language Used

Python (3.7)

IDE Used

Spyder from Anaconda Navigator

Libraries Used

pandas, numpy, plotly, seaborn, tkinter, matplotlib, pymysql

Database Used:

MySQL through XAMPP Server

Overview of the Project:

This project offers various features that have been implemented using python and its concepts of machine learning.
The features implemented are as follows:

1.Sales Prediction
2.MIS Report Generation
3.Chiller Data Analysis
4.Cross Connects Prediction
5.Compressor Wise Data Analysis

I tried to solve some of the problems faced by the members of the NOC Team and the Facility team by writing some basic codes that could yield them some of the results and save their time to focus on some other critical tasks which required some prioritized attention.

Using the python libraries like pandas, numpy, plotly, seaborn, tkinter, matplotlib, pymysql I have made a simple UI(User Interface) through which a user can create an account of their own and login into the UI to explore the features provided.

After the user logs in, he/she can see the functions available for use. Once the user clicks on that function button. The next thing will be either a dialogue box appears with some more options or the function is simply executed as asked.
The features present in the interface are as follows :

1. Sales Prediction

This feature enables the user to predict the sales that would occur in the further years down the line using the linear regression algorithm
There is a scatter plot given for visualization in terms of rise or fall of sales occurring in a particular year

2. MIS Report Generation

This function saves the time of the user by bifurcating a .csv file consisting of data of numerous client into multiple .csv files with the customer name and their respective data. This feature saves the time of user spent in manually filtering data and letting them focus on the work that has higher priority.

3. Chiller Data Analysis

This function enables the user to analyse the data of a chiller in any particular facility through the data in a .csv file by plotting the respective graphs based on some critical parameters.

4. Cross Connects Prediction

This feature enables the user to predict the number of cross connect request that would be coming in the further years down the line using the linear regression algorithm.
There is a scatter plot given for visualization in terms of rise or fall of the number cross connect requests coming in a particular year

5. Compressor Wise Data Analysis

This feature enables the user to compare multiple parameters of one particular compressor in one particular chiller in a single graph. This feature increases the visualization of the user by giving them the opportunity to compare multiple parameters of a chiller in one single place

<u>Machine Learning Concept Calculations and Statistics</u>:

Cross Connect Prediction Calculations

Year	Number of Cross Connect	x-mean(x)	y-mean(y)	(x-mean(x))^2	(y-mean(y))*(x-mean(x))	Predicted Values
2016	258	-1.5	-311	2.25	466.5	330.175
2017	569	-0.5	8.5	0.25	-4.25	483.725
2018	696	0.5	135.5	0.25	67.75	637.275
2019	719	1.5	158.5	2.25	237.75	790.825
Mean= 2017.5	Mean= 560.5			Sum=5	Sum= 767.5	

m= 767.75/5 = 153.55
y=mx+c
560.5=153.55*(2017.5)+c
Therefore, c=-309226.625

Sales Prediction Calculations

Year	Number of Clients	x-mean(x)	y-mean(y)	(x-mean(x))^2	(y-mean(y))*(x-mean(x))	Predicted Values
2017	119	-1	4.34	1	-4.34	118.99
2018	75	0	-39.66	0	-39.66	114.66
2019	150	1	35.34	1	35.34	110.33
Mean= 2018	Mean= 114.66			Sum=2	Sum= -8.66	

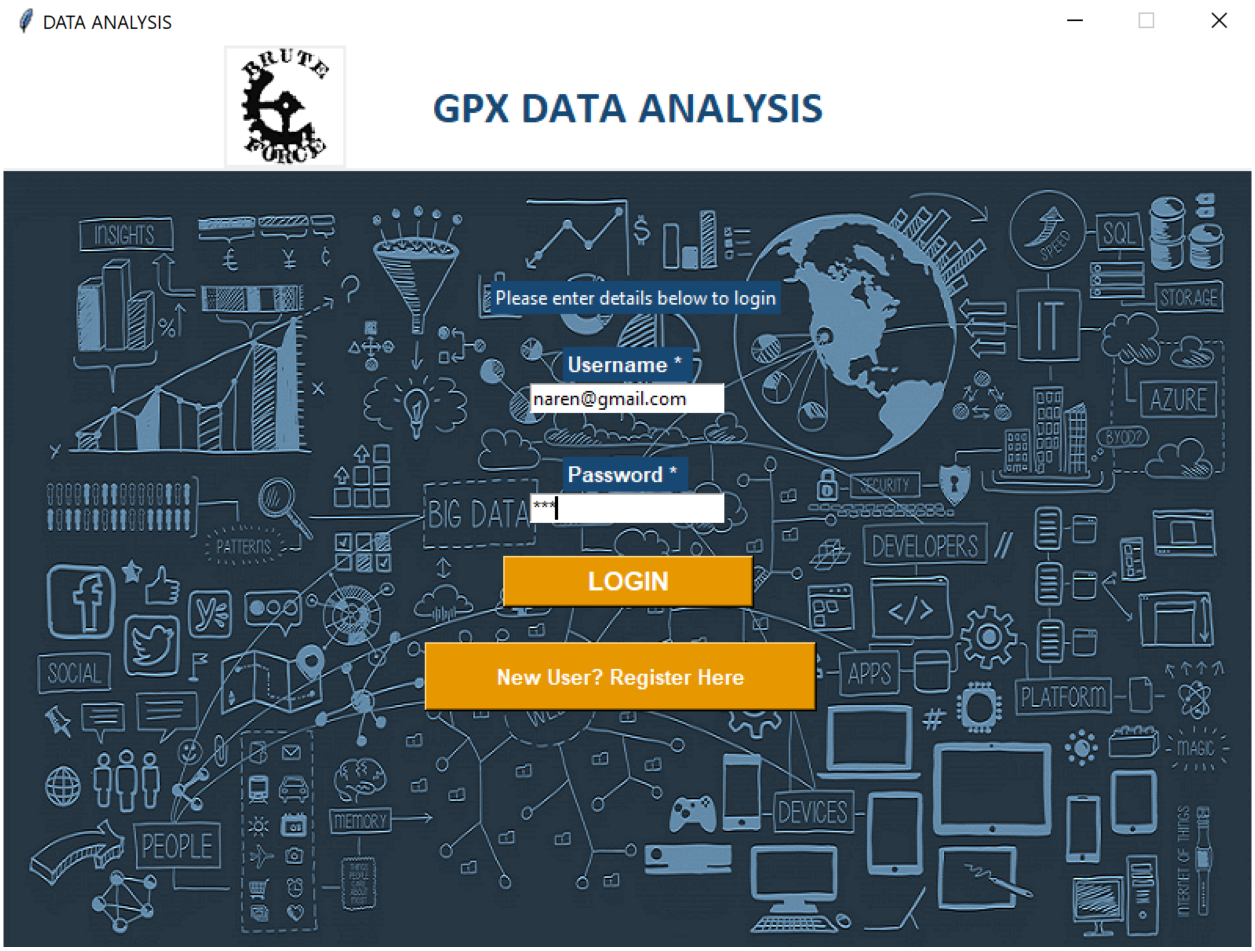
m= (-8.66)/2 = -4.33
y=mx+c
714.66=(-4.33)*(2018)+c
Therefore, c=8852.6

Source Code Subsections with their respective outputs:

1) Login and Interface to invoke the created function:

```
def main_screen():
    global screen, username_verify, password_verify
    screen = Tk()
    username_verify = StringVar()
    password_verify = StringVar()
    screen.title("DATA ANALYSIS")
    adjustWindow(screen)
    img2 = ImageTk.PhotoImage(Image.open("bigdata.png"))
    d1 = Label(image= img2)
    d1.place(x =-1,y = 78)


 Label(screen,text="GPX DATA
ANALYSIS",width="55",height="2",font=("Calibri",22,'bold'),bg='white',fg='#17487
3').pack()
    Label(screen, text="Please enter details below to login", bg='#174873',
fg='white').place(x=312,y=150)
    Label(screen, text="Username * ", font=("Open Sans", 10, 'bold'),
bg='#174873', fg='white').place(x=358,y=192)
    Entry(screen, textvar=username_verify).place(x=337,y=215)
    Label(screen, text="Password * ", font=("Open Sans", 10, 'bold'),
bg='#174873', fg='white').place(x=358,y=262)
    Entry(screen, textvar=password_verify, show="*").place(x=337,y=285)
    Button(screen, text="LOGIN", bg="#e79700", width=15, height=1, font=("Open
Sans", 13, 'bold'), fg='white', command=login_verify).place(x=320,y=325)
    Button(screen, text="New User? Register Here", height="2", width="30",
bg='#e79700', font=("Open Sans", 10, 'bold'), fg='white',
command=register).place(x=270,y=380)
```

```
img1 = ImageTk.PhotoImage(Image.open("bft.png"))
c1 = Label(image= img1)
c1.place(x =142,y = 0)
screen.mainloop()
```



DATA ANALYSIS

GPX DATA ANALYSIS

Please enter details below to login

Username *

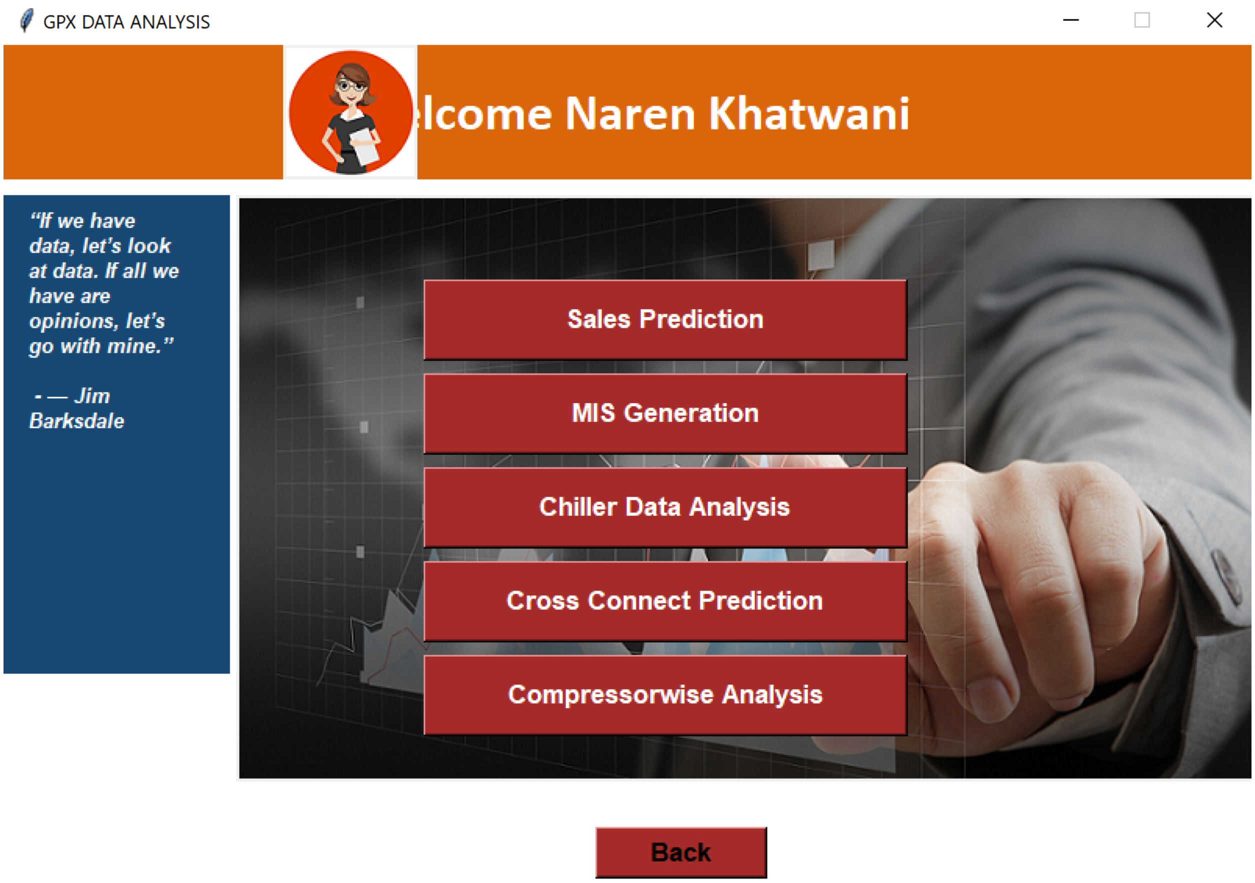
naren@gmail.com

Password *

***

```
def welcome_page(registration_info):
    global screen2
    screen2 = Toplevel(screen)
    screen2.title("GPX DATA ANALYSIS")
    adjustWindow(screen2) # configuring the window
    Label(screen2, text="Welcome " + registration_info[0][1], width='47',
height="2", font=("Calibri", 25, 'bold'), fg='white', bg='#d9660a').place(x=0, y=0)
    Label(screen2, text="", bg='#174873', width='20', height='20').place(x=0, y=96)
    Message(screen2, text='“If we have data, let’s look at data. If all we have are
opinions, let’s go with mine.”\n\n - — Jim Barksdale', width='100',
font=("Helvetica", 10, 'bold', 'italic'), fg='white', bg='#174873', anchor =
CENTER).place(x=10, y=100)

    photo1 = PhotoImage(file="analy.png") # opening right side image - Note: If
image is in same folder then no need to mention the full path
    label1 = Label(screen2, image=photo1, text="") # attaching image to the label
    label1.place(x=150, y=96)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image
to display image in label

    photo1 = PhotoImage(file="pay.png") # opening right side image - Note: If
image is in same folder then no need to mention the full path
    label1 = Label(screen2, image=photo1, text="") # attaching image to the label
    label1.place(x=180, y=0)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image
to display image in label
    Button(screen2, text='Sales Prediction', width=30,height=2, font=("Open
Sans", 13, 'bold'),command=mlwindow, bg='brown', fg='white').place(x=270,
y=150)
    Button(screen2, text='MIS Generation', width=30,height=2, font=("Open Sans",
13, 'bold'),command=misgeneration, bg='brown', fg='white').place(x=270, y=210)
    Button(screen2, text='Chiller Data Analysis', width=30,height=2, font=("Open
Sans", 13, 'bold'), bg='brown', fg='white',command=next_page).place(x=270,
y=270)
```

```
   Button(screen2, text='Cross Connect Prediction', width=30,height=2,
font=("Open Sans", 13, 'bold'), bg='brown',
fg='white',command=mlwindow2).place(x=270, y=330)
   Button(screen2, text='Compressorwise Analysis', width=30,height=2,
font=("Open Sans", 13, 'bold'), bg='brown',
fg='white',command=companalysis).place(x=270, y=390)
   Button(screen2, text='Back', width=10, font=("Open Sans", 13, 'bold'),
bg='brown', fg='black',command=screen2.destroy).place(x=380, y=500)
```



lcome Naren Khatwani

“If we have data, let’s look at data. If all we have are opinions, let’s go with mine.”

- — Jim Barksdale

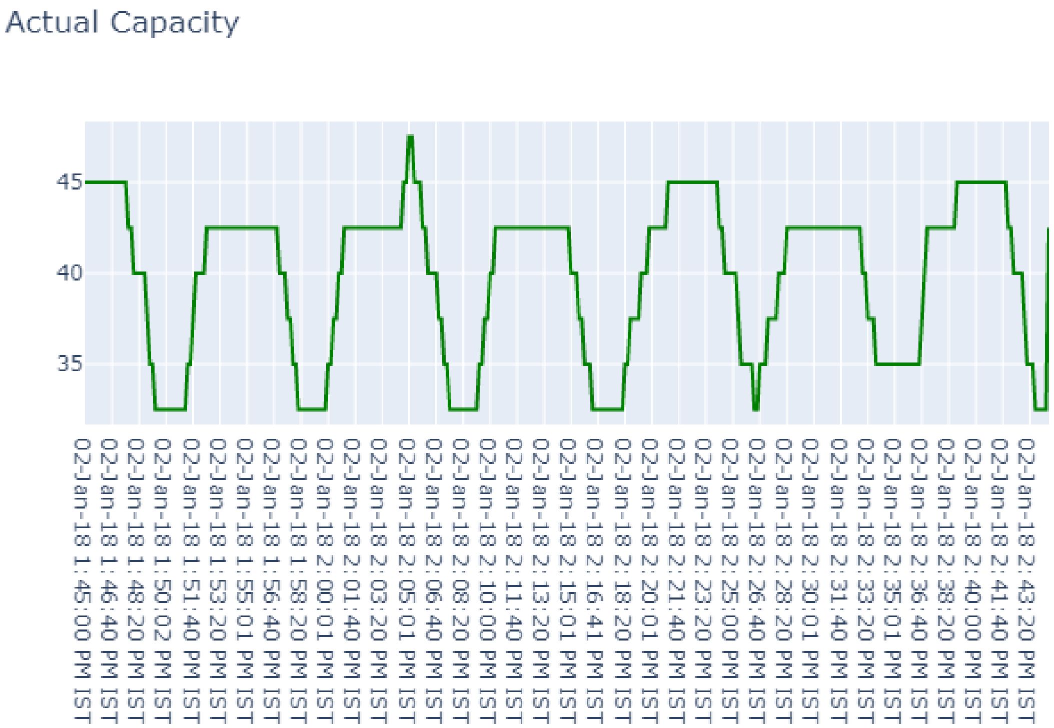
Function to adjust the geometry of a specific screen:

```
def adjustWindow(window):
    w = 800  # width for the window size
    h = 600  # height for the window size
    ws = screen.winfo_screenwidth()  # width of the screen
    hs = screen.winfo_screenheight()  # height of the screen
    x = (ws/2) - (w/2)  # calculate x and y coordinates for the Tk window
    y = (hs/2) - (h/2)
    window.geometry('%dx%d+%d+%d' % (w, h, x, y))  # set the dimensions of the screen and where it is placed
    window.resizable(False, False)    # disabling the resize option for the window
    window.configure(background='white')    # making the background white of the window
```

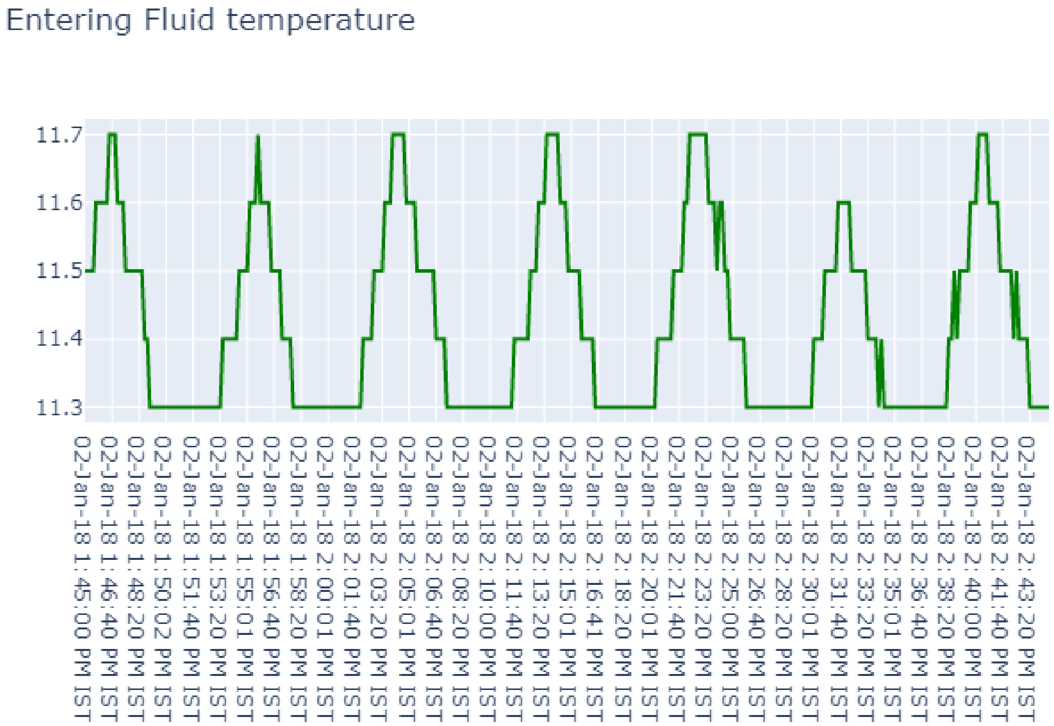
The above function adjusts the geometry of a particular screen created using tkinter

Graphical Illustrations of the Chiller Data

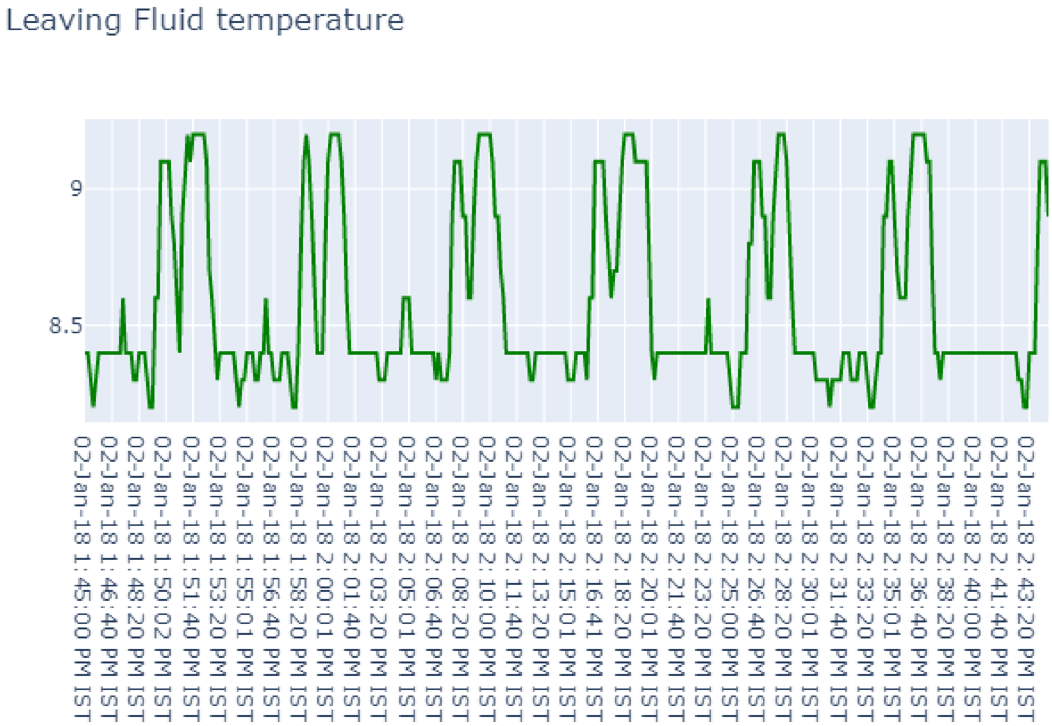
```
def question_1():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual
Capacity'], name="COND REF PRESS COMP1",line_color='green'))
    fig1.update_layout(title_text='Actual Capacity',xaxis_rangeslider_visible=True)


    fig1.show()
    plot(fig1)
```



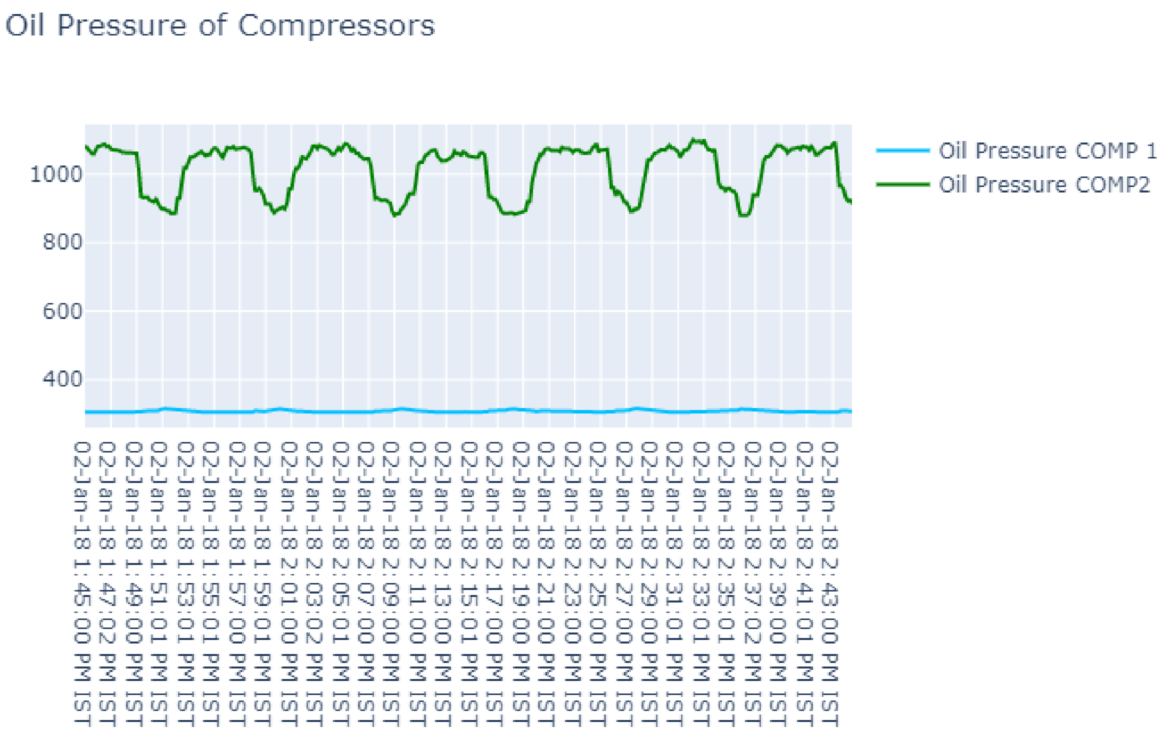
```
def question_2():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid temp'], name="COND REF PRESS COMP1",line_color='green'))
   fig1.update_layout(title_text='Entering Fluid temperature',xaxis_rangeslider_visible=True)


   fig1.show()
   plot(fig1)
```



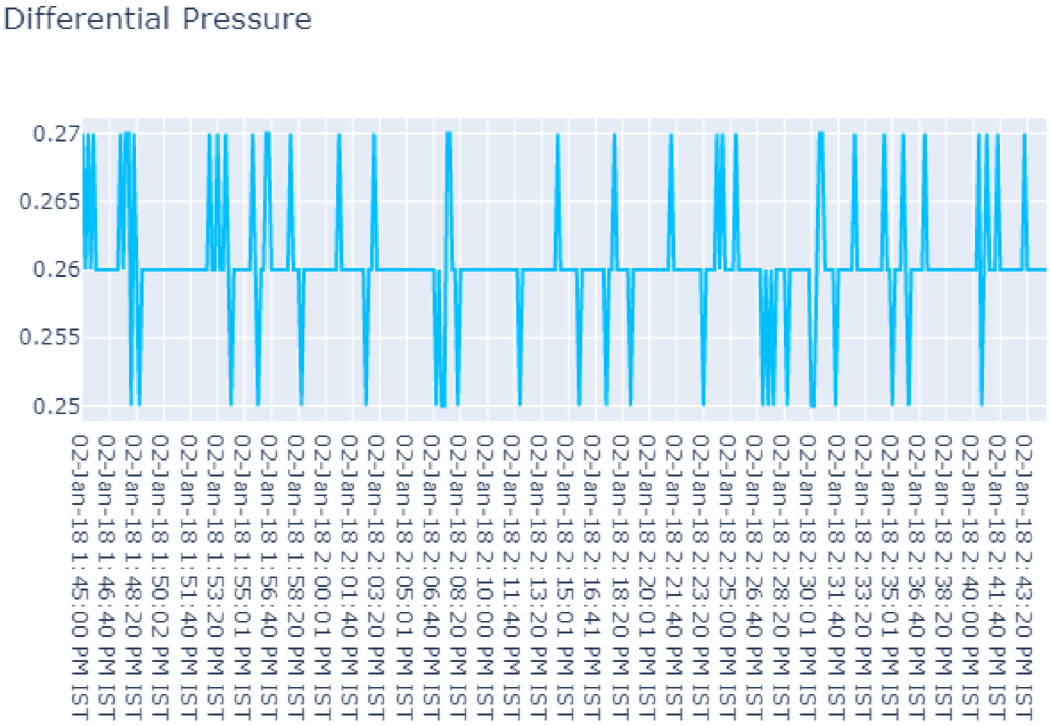
```
def question_3():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid
temp'], name="COND REF PRESS COMP1",line_color='green'))
   fig1.update_layout(title_text='Leaving Fluid
temperature',xaxis_rangeslider_visible=True)


   fig1.show()
   plot(fig1)
```



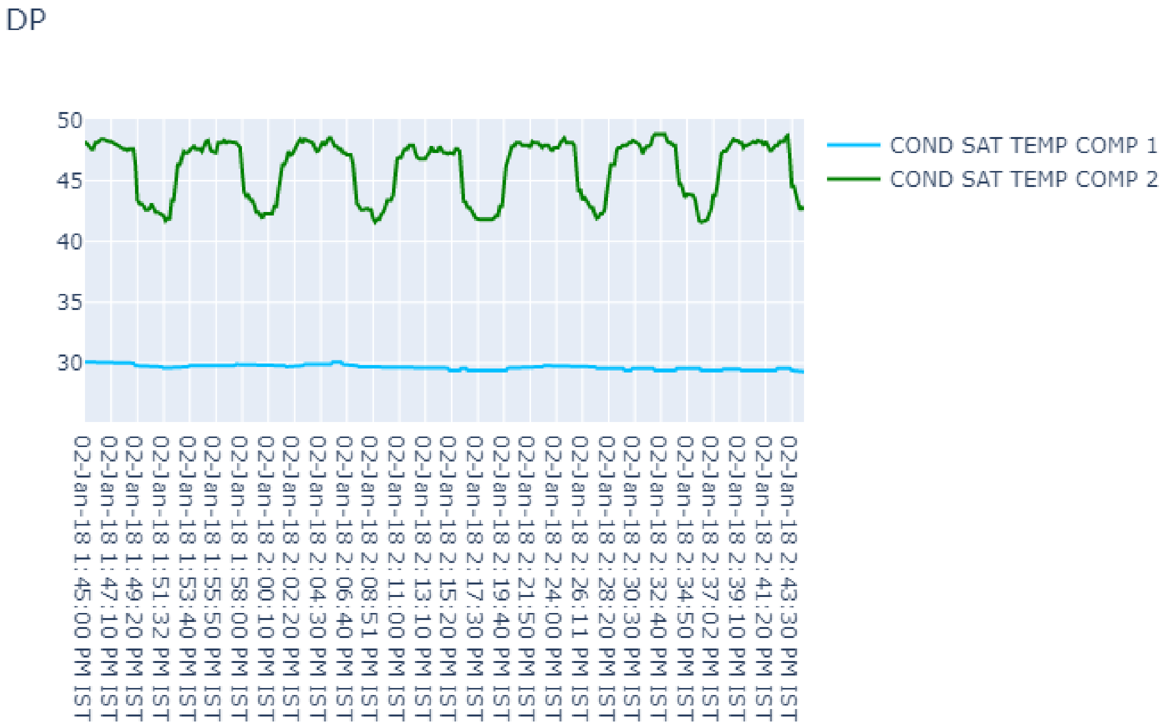
```
def question_4():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure
COMP1'], name="Oil Pressure COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure
COMP2'], name="Oil Pressure COMP2",line_color='green'))

    fig1.update_layout(title_text='Oil Pressure of
Compressors',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



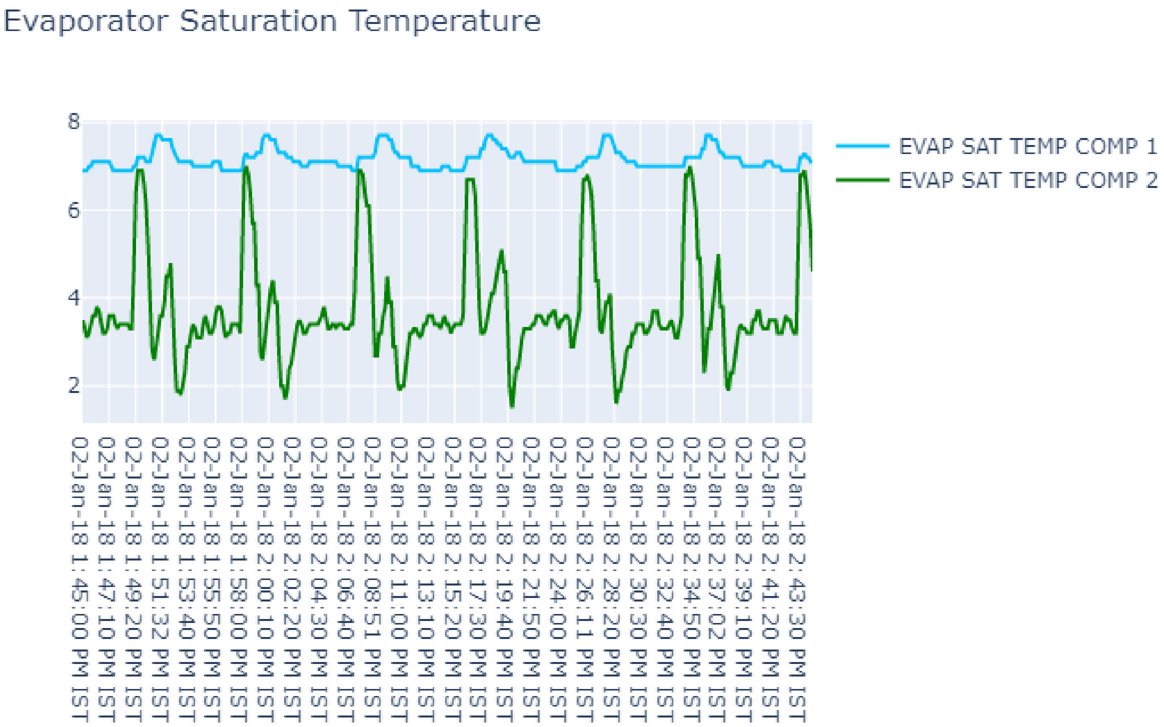
```
def question_5():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'],
name="DP",line_color='deepskyblue'))

    fig1.update_layout(title_text='Differential Pressure',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



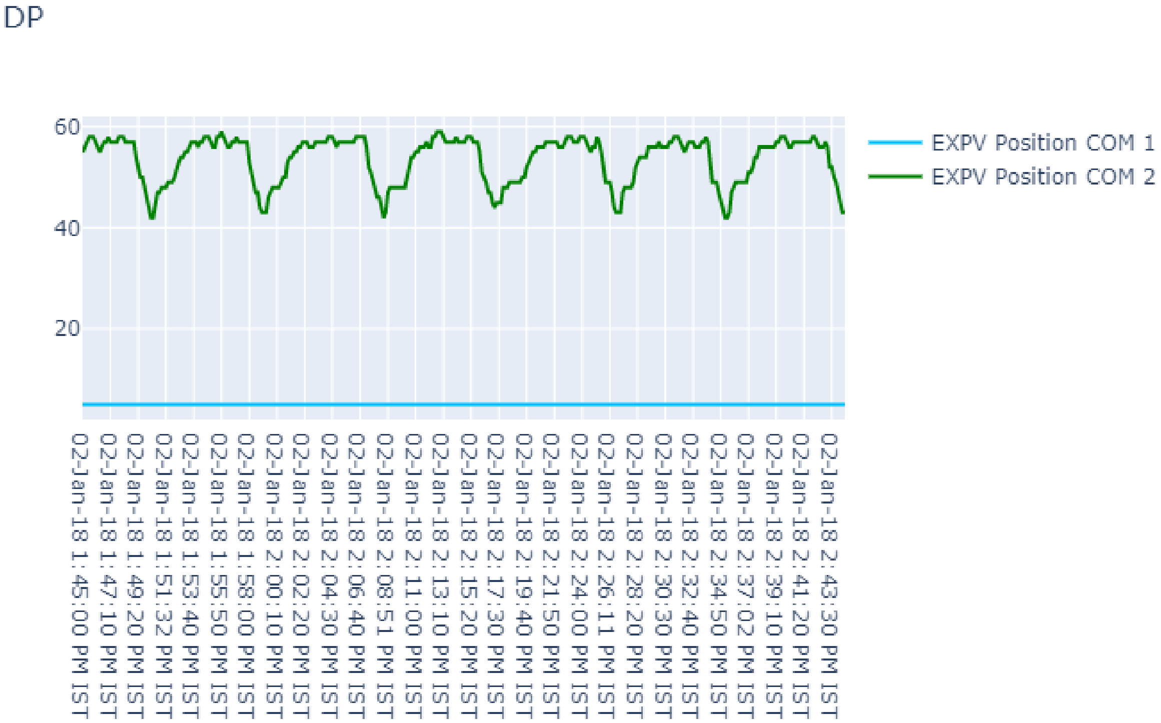
```
def question_6():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT TEMP COMP 1'], name="COND SAT TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT TEMP COMP 2'], name="COND SAT TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



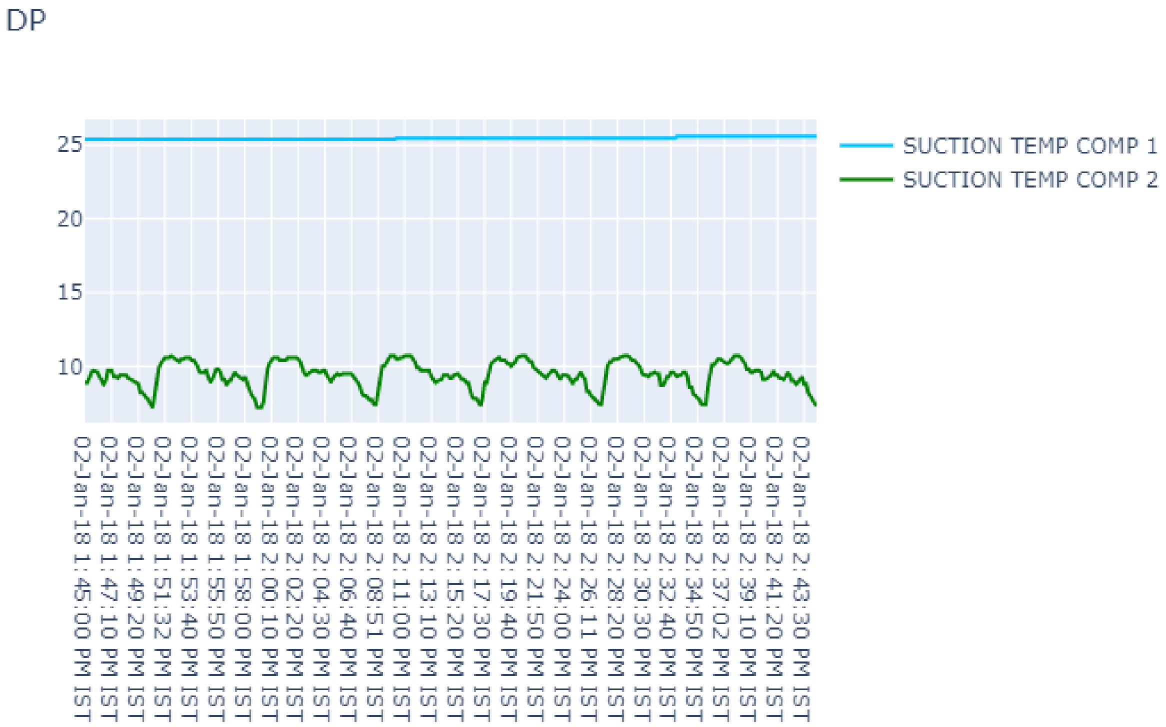
```
def question_7():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT TEMP COM1'], name="EVAP SAT TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT TEMP COM2'], name="EVAP SAT TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='Evaporator Saturation Temperature',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



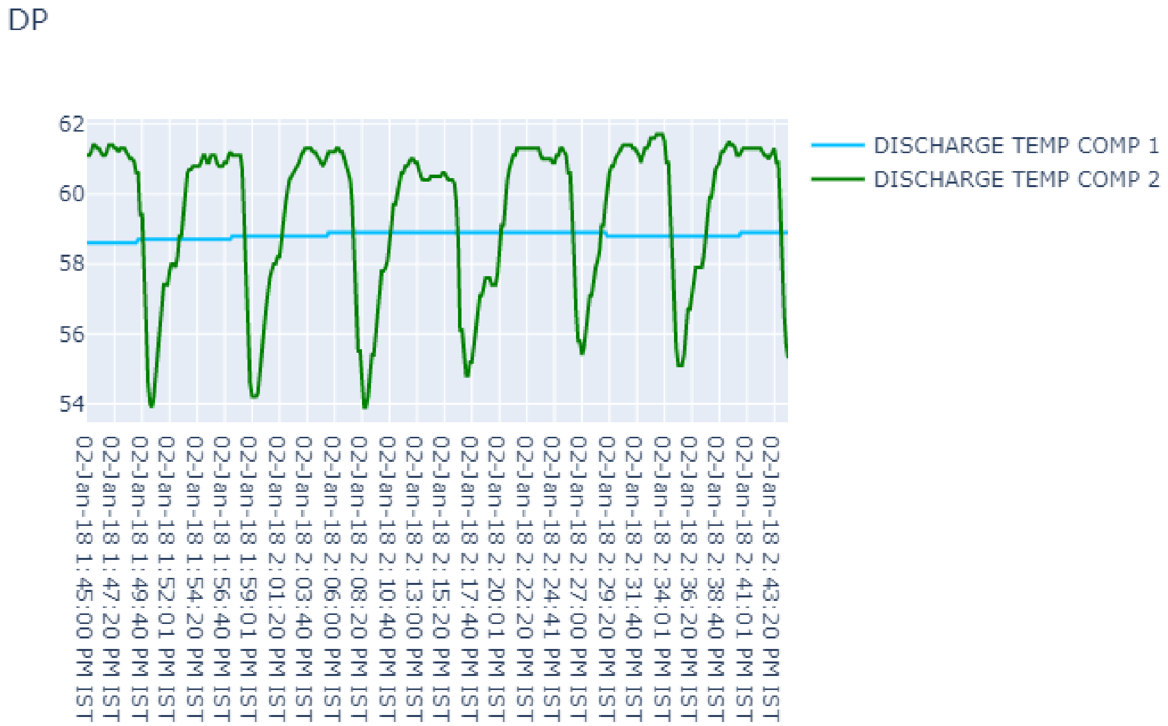
```
def question_8():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position COM 1'], name="EXPV Position COM 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position COM 2'], name="EXPV Position COM 2",line_color='green'))

   fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)
```



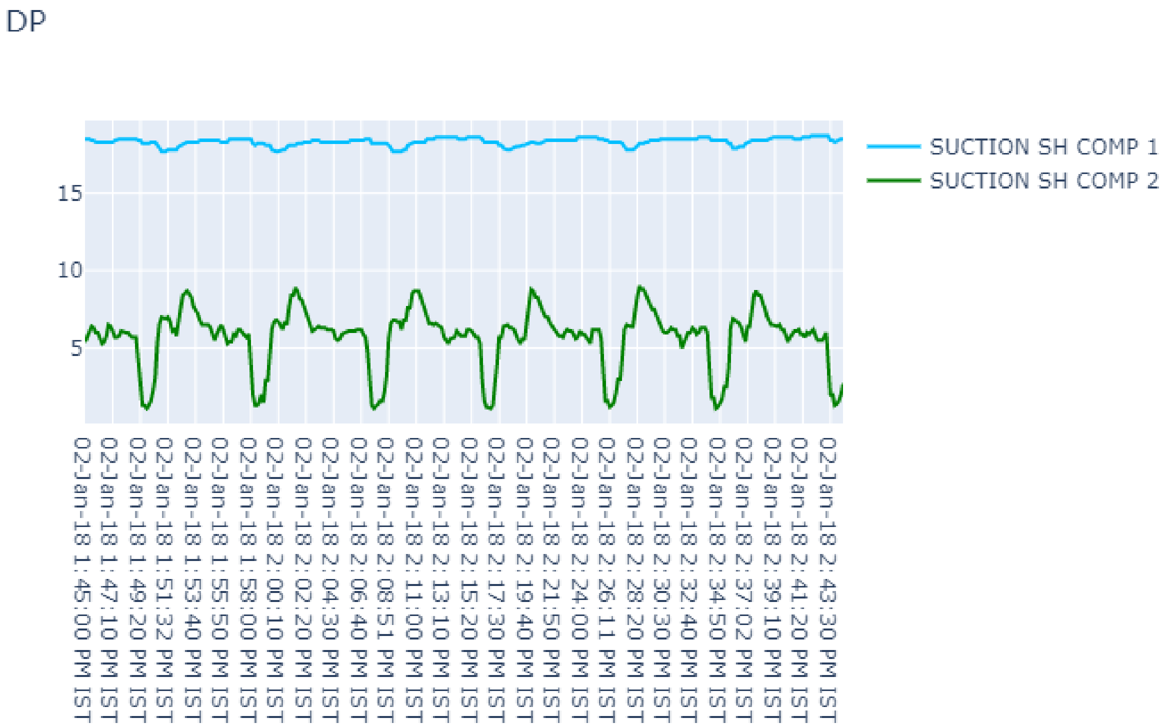
```
def question_9():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION
TEMP COMP 1'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION
TEMP COMP 2'], name="SUCTION TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



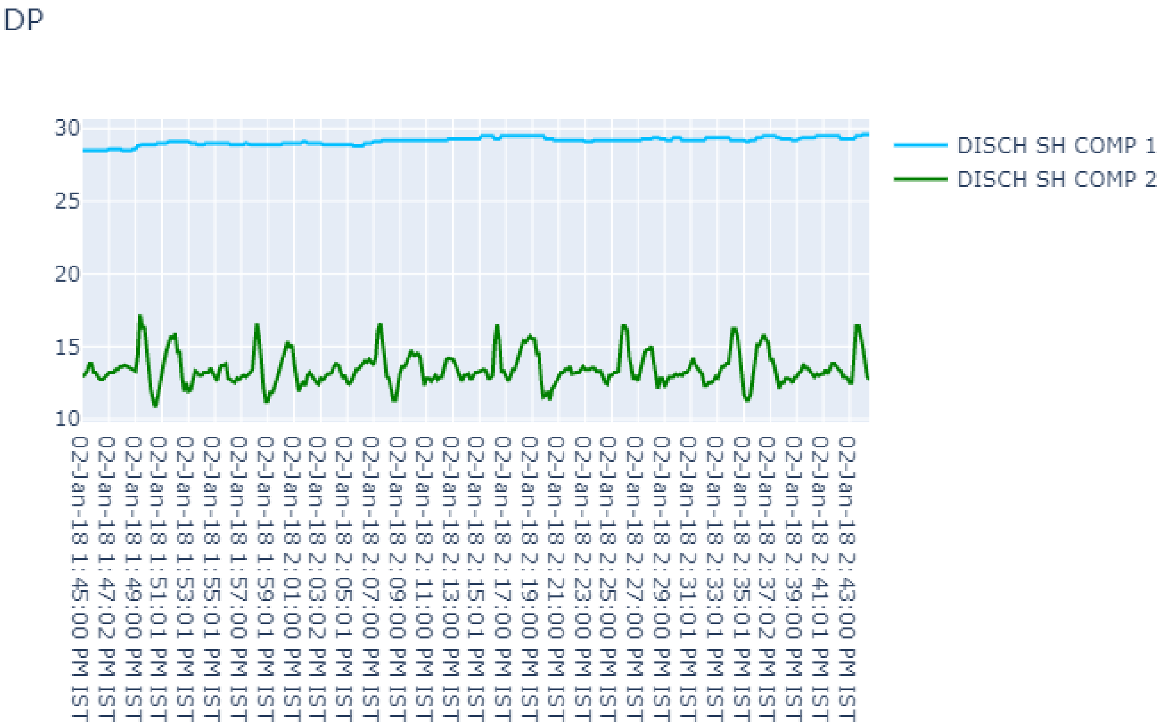
```
def question_10():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE
TEMP COMP 1'], name="DISCHARGE TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE
TEMP COMP 2'], name="DISCHARGE TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



```
def question_11():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH
COMP 1'], name="SUCTION SH COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH
COMP 2'], name="SUCTION SH COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



```
def question_12():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH SH
COMP 1'], name="DISCH SH COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH SH
COMP 2'], name="DISCH SH COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```



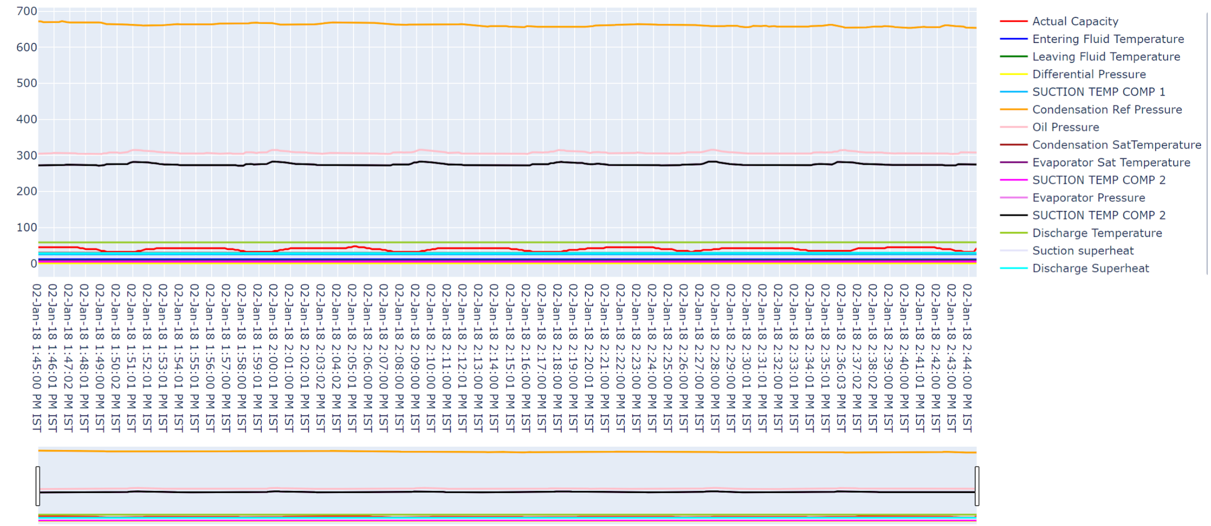
```
def question_13():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF
PRESS COMP1'], name="COND REF PRESS COMP1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF
PRESS COMP2'], name="COND REF PRESS COMP2",line_color='green'))

    fig1.update_layout(title_text='Condensation Ref Pressure of
Compressors',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

def question_14():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP1'], name="EVAPORATIVE PRESSURE COMPRESSOR 1",line_color='green'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP2'], name="EVAPORATIVE PRESSURE COMPRESSOR
1",line_color='deepskyblue'))
   fig1.update_layout(title_text='Evaporator Pressure
Compressorwise',xaxis_rangeslider_visible=True)


   fig1.show()
   plot(fig1)
```

Compressorwise Data Analysis

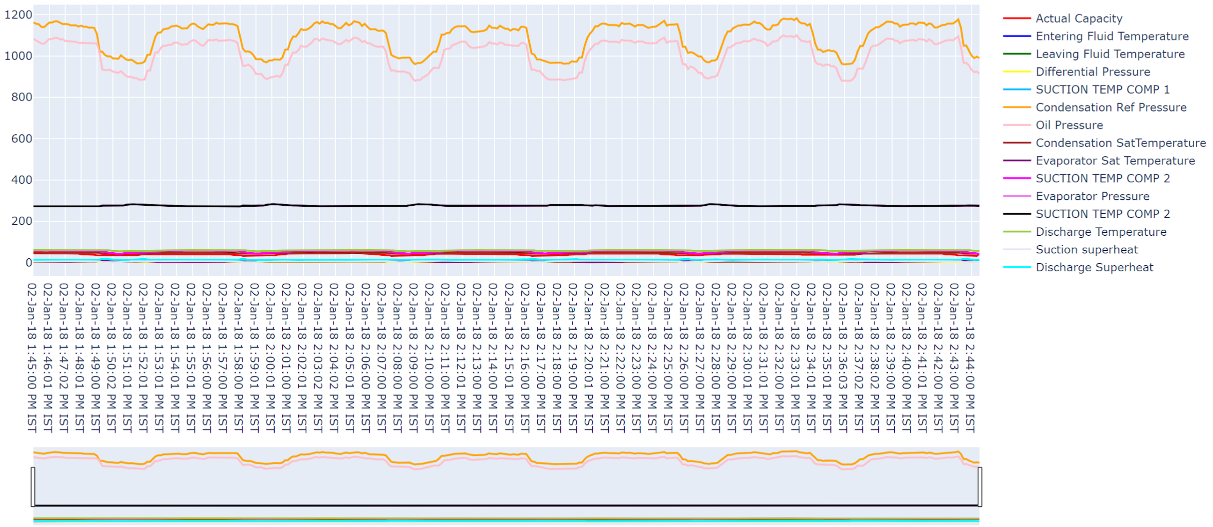
```
def comp1():

    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual Capacity'], name="Actual Capacity",line_color='red'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid temp'], name="Entering Fluid Temperature",line_color='blue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid temp'], name="Leaving Fluid Temperature",line_color='green'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'], name="Differential Pressure",line_color='yellow'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION TEMP COMP 1'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF PRESS COMP1'], name="Condensation Ref Pressure ",line_color='orange'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure COMP1'], name="Oil Pressure",line_color='pink'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT TEMP COMP 1'], name="Condensation SatTemperature",line_color='brown'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT TEMP COM1'], name="Evaporator Sat Temperature",line_color='purple'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position COM 1'], name="SUCTION TEMP COMP 2",line_color='magenta'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS COMP1'], name="Evaporator Pressure",line_color='violet'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS COMP1'], name="SUCTION TEMP COMP 2",line_color='black'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE TEMP COMP 1'], name="Discharge Temperature",line_color='yellowgreen'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH COMP 1'], name="Suction superheat",line_color='lavender'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH     SH COMP 1'], name="Discharge Superheat",line_color='cyan'))

    fig1.update_layout(title_text='COMPRESSOR 1',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)
```

```
def comp2():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual
Capacity'], name="Actual Capacity",line_color='red'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid
temp'], name="Entering Fluid Temperature",line_color='blue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid
temp'], name="Leaving Fluid Temperature",line_color='green'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'],
name="Differential Pressure",line_color='yellow'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION
TEMP COMP 2'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF
PRESS COMP2'], name="Condensation Ref Pressure ",line_color='orange'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure
COMP2'], name="Oil Pressure",line_color='pink'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT
TEMP COMP 2'], name="Condensation SatTemperature",line_color='brown'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT
TEMP COM2'], name="Evaporator Sat Temperature",line_color='purple'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position
COM 2'], name="SUCTION TEMP COMP 2",line_color='magenta'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP2'], name="Evaporator Pressure",line_color='violet'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP2'], name="SUCTION TEMP COMP 2",line_color='black'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE
TEMP COMP 2'], name="Discharge Temperature",line_color='yellowgreen'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH
COMP 2'], name="Suction superheat",line_color='lavender'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH    SH
COMP 2'], name="Discharge Superheat",line_color='cyan'))

   fig1.update_layout(title_text='COMPRESSOR 2',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)
```

COMPRESSOR 1

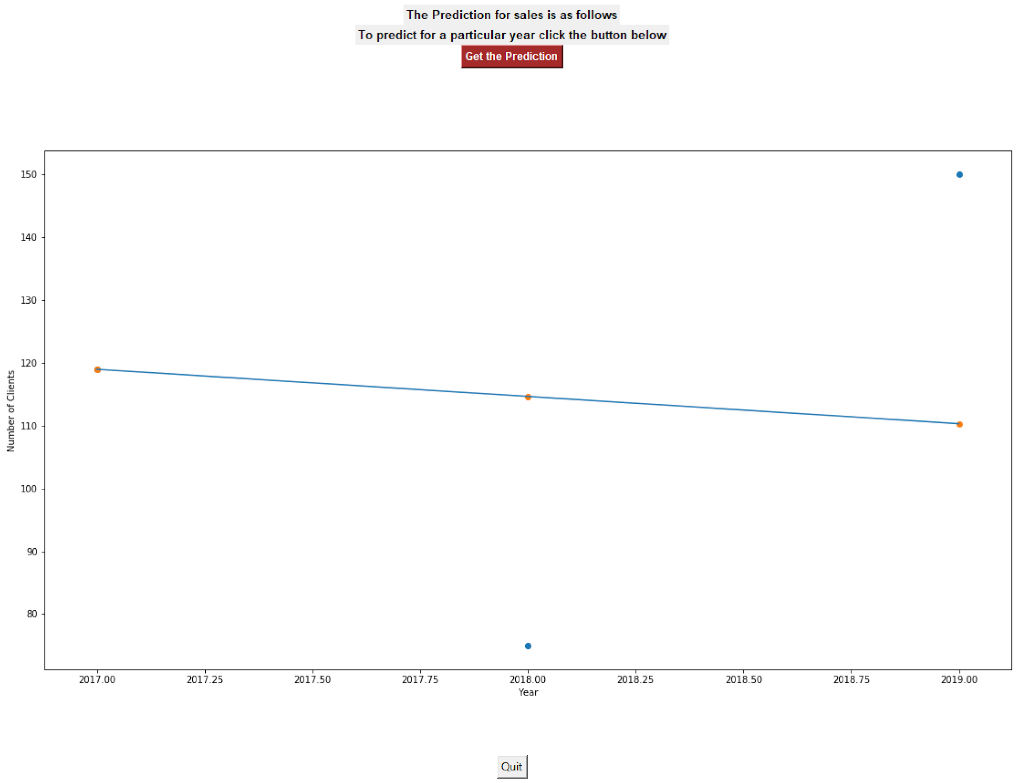


COMPRESSOR 2



Machine Learning Sales Prediction (Number of Clients) and Cross Connects Prediction:

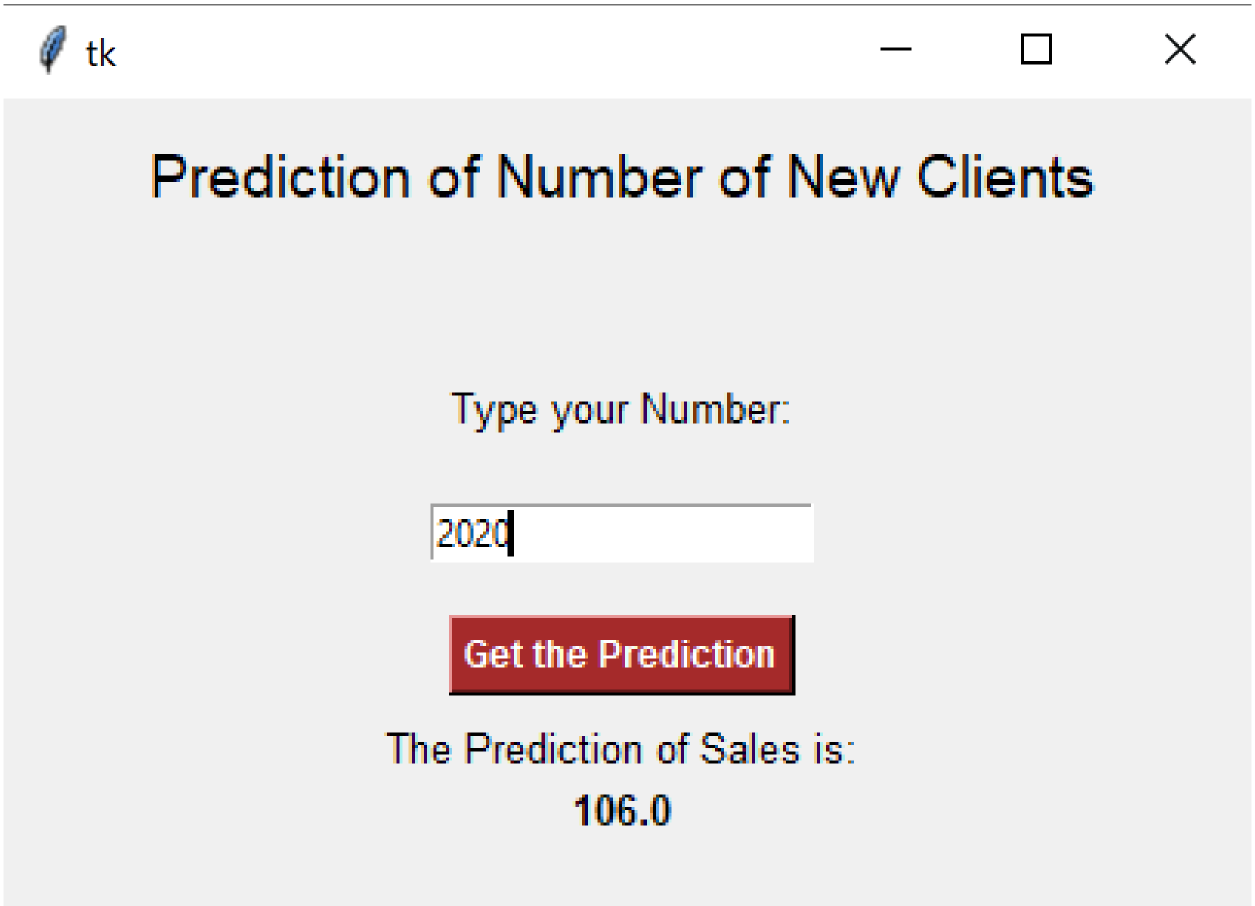
```
def mlwindow():
    global screen55
    screen55 = Tk()
    screen55.title('Question-1')
    adjustWindow1(screen55)

    Label(screen55, text='The Prediction for sales is as follows' ,font=('helvetica', 10,
'bold')).pack()
    Label(screen55, text='To predict for a particular year click the button below'
,font=('helvetica', 10, 'bold')).pack()
    Button(screen55,text='Get the Prediction',command=mlpredict, bg='brown', fg='white',
font=('helvetica', 9, 'bold')).pack()
    fig, ax = plt.subplots()

    data=np.array([
    [2017,119],
    [2018,75],
    [2019,150]])

    x,y=data.T
    plt.ylabel("Number of Clients")
    plt.xlabel("Year")
    plt.scatter(x,y)


    data1=np.array([
    [2017,118.99],
    [2018,114.66],
    [2019,110.33]])

    x,y=data1.T
    plt.ylabel("Number of Clients")
    plt.xlabel("Year")
    plt.scatter(x,y)
    plt.plot(x,y)
    canvas = FigureCanvasTkAgg(fig, screen55)
    canvas.draw()
    canvas.get_tk_widget().pack(side = TOP, fill = BOTH, expand = True)

    button = Button(screen55, text="Quit", command=screen55.destroy)
    button.pack(side=BOTTOM)
```



The Prediction for sales is as follows

To predict for a particular year click the button below

```
def mlpredict():
    import tkinter as tk

    root= tk.Tk()

    canvas1 = tk.Canvas(root, width = 400, height = 300,  relief = 'raised')
    canvas1.pack()

    label1 = tk.Label(root, text='Prediction of Number of New Clients')
    label1.config(font=('helvetica', 14))
    canvas1.create_window(200, 25, window=label1)

    label2 = tk.Label(root, text='Type your Number:')
    label2.config(font=('helvetica', 10))
    canvas1.create_window(200, 100, window=label2)

    entry1 = tk.Entry (root)
    canvas1.create_window(200, 140, window=entry1)

    def getSquareRoot ():

        x1 = entry1.get()
        x2=(float(x1)*(-4.33))+(8852.6)
        x3=str(x2)
        label3 = tk.Label(root, text= 'The Prediction of Sales" is:',font=('helvetica', 10))
        canvas1.create_window(200, 210, window=label3)

        label4 = tk.Label(root, text=x3 ,font=('helvetica', 10, 'bold'))
        canvas1.create_window(200, 230, window=label4)

    button1 = tk.Button(root,text='Get the Prediction', command=getSquareRoot,
bg='brown', fg='white', font=('helvetica', 9, 'bold'))

    canvas1.create_window(200, 180, window=button1)

    root.mainloop()
```



Prediction of Number of New Clients

Type your Number:

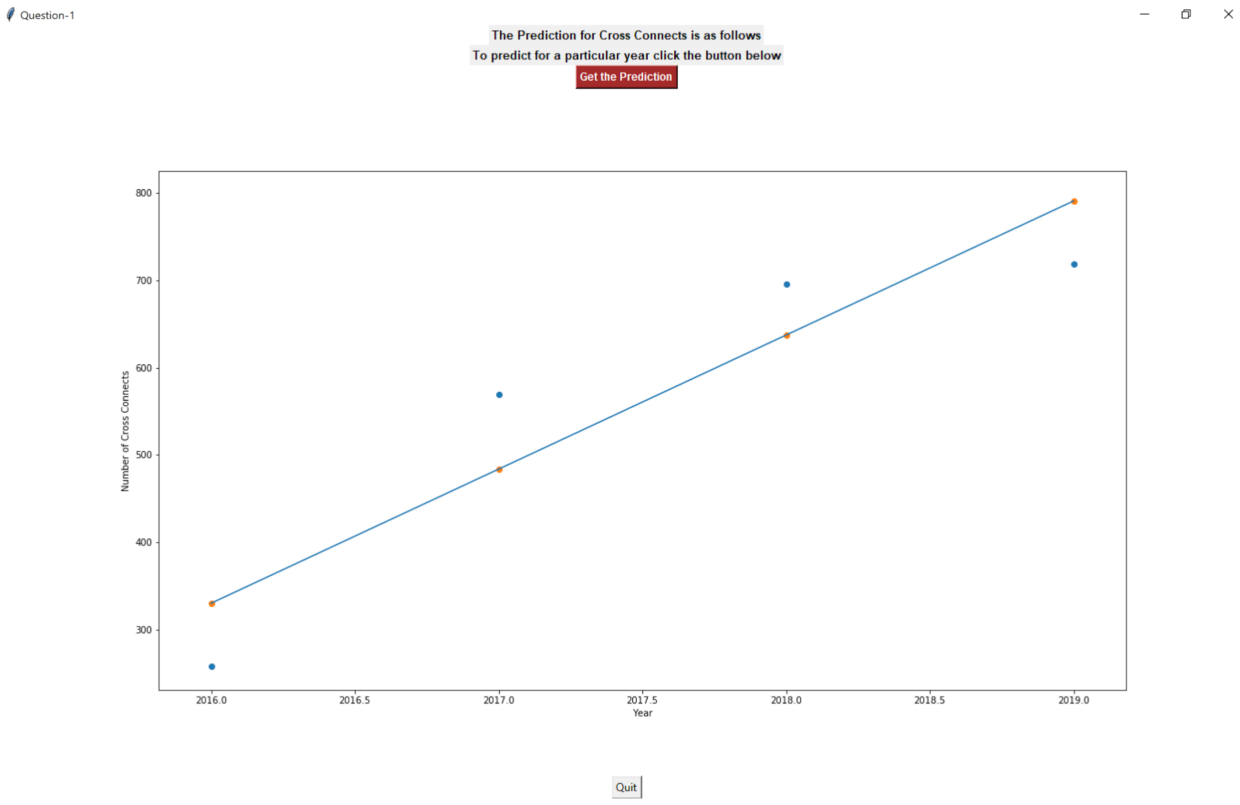
2020

The Prediction of Sales is:

106.0

#cross connect prediction

```
def mlwindow2():
    global screen65
    screen65 = Tk()
    screen65.title('Question-1')
    adjustWindow1(screen65)

    Label(screen65, text='The Prediction for Cross Connects is as follows'
,font=('helvetica', 10, 'bold')).pack()
    Label(screen65, text='To predict for a particular year click the button below'
,font=('helvetica', 10, 'bold')).pack()
    Button(screen65,text='Get the Prediction',command=mlpredict2, bg='brown',
fg='white', font=('helvetica', 9, 'bold')).pack()
    fig, ax = plt.subplots()

    data=np.array([
    [2016,258],
    [2017,569],
    [2018,696],
    [2019,719]])

    x,y=data.T
    plt.ylabel("Number of Cross Connects")
    plt.xlabel("Year")
    plt.scatter(x,y)



    data1=np.array([
    [2016,330.175],
    [2017,483.725],
    [2018,637.275],
    [2019,790.825]])

    x,y=data1.T
    plt.ylabel("Number of Cross Connects")
    plt.xlabel("Year")
    plt.scatter(x,y)
    plt.plot(x,y)
    canvas = FigureCanvasTkAgg(fig, screen65)
    canvas.draw()
    canvas.get_tk_widget().pack(side = TOP, fill = BOTH, expand = True)

    button = Button(screen65, text="Quit", command=screen55.destroy)
    button.pack(side=BOTTOM)
```

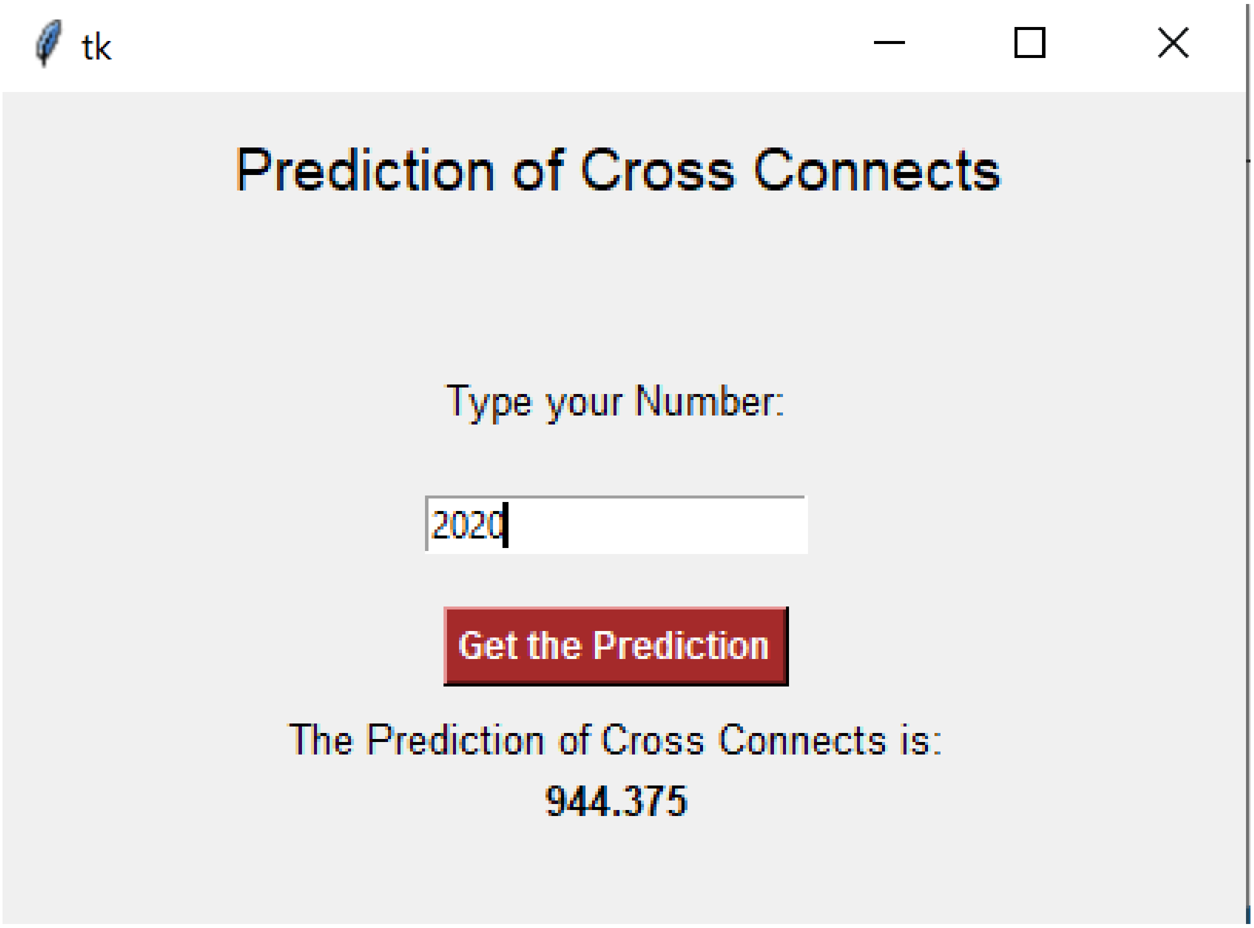


Question-1

The Prediction for Cross Connects is as follows

To predict for a particular year click the button below

```
def mlpredict2():
    import tkinter as tk

    root1= tk.Tk()

    canvas1 = tk.Canvas(root1, width = 400, height = 300,  relief = 'raised')
    canvas1.pack()

    label1 = tk.Label(root1, text='Prediction of Cross Connects')
    label1.config(font=('helvetica', 14))
    canvas1.create_window(200, 25, window=label1)

    label2 = tk.Label(root1, text='Type your Number:')
    label2.config(font=('helvetica', 10))
    canvas1.create_window(200, 100, window=label2)

    entry1 = tk.Entry (root1)
    canvas1.create_window(200, 140, window=entry1)

    def getSquareRoot2 ():

        x1 = entry1.get()
        x2=(float(x1)*153.55)-(309226.625)
        x3=str(x2)
        label3 = tk.Label(root1, text= 'The Prediction of Cross Connects"
is:',font=('helvetica', 10))
        canvas1.create_window(200, 210, window=label3)

        label4 = tk.Label(root1, text=x3 ,font=('helvetica', 10, 'bold'))
        canvas1.create_window(200, 230, window=label4)

    button1 = tk.Button(root1,text='Get the Prediction', command=getSquareRoot2,
bg='brown', fg='white', font=('helvetica', 9, 'bold'))

    canvas1.create_window(200, 180, window=button1)

    root1.mainloop()
```



Prediction of Cross Connects

Type your Number:

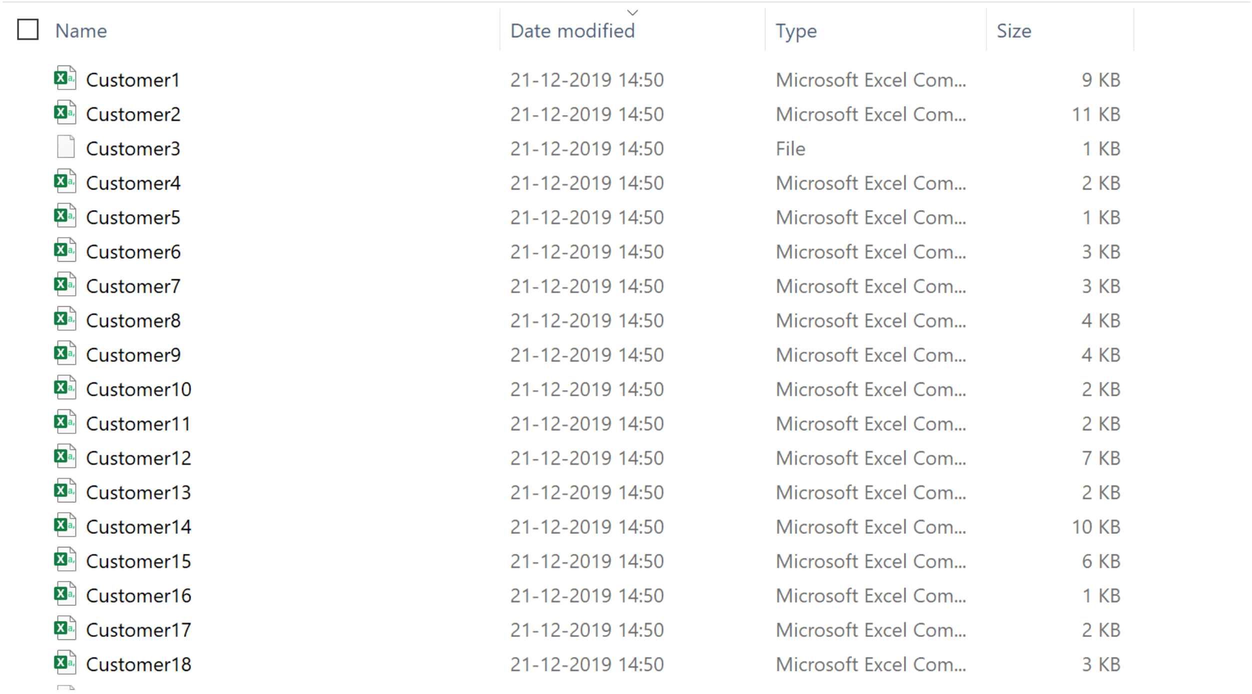
2020

The Prediction of Cross Connects is:

944.375

MIS Report Generation:

```
data_naren=pd.read_csv("sunilsir.csv",usecols=[0,1,7,8,11,12,13,14])
data_naren1=data_naren[(data_naren['Organization']=='Customername')]
data_naren2=data_naren[(data_naren['Organization']=='Customername')]
data_naren3=data_naren[(data_naren['Organization']=='Customername')]
data_naren4=data_naren[(data_naren['Organization']=='Customername')]
data_naren5=data_naren[(data_naren['Organization']=='Customername')]
data_naren6=data_naren[(data_naren['Organization']=='Customername')]
data_naren7=data_naren[(data_naren['Organization']=='Customername')]
data_naren8=data_naren[(data_naren['Organization']=='Customername')]
data_naren9=data_naren[(data_naren['Organization']=='Customername')]
data_naren10=data_naren[(data_naren['Organization']=='Customername')]
data_naren11=data_naren[(data_naren['Organization']=='Customername')]
data_naren12=data_naren[(data_naren['Organization']=='Customername')]
data_naren13=data_naren[(data_naren['Organization']=='Customername')]
data_naren14=data_naren[(data_naren['Organization']=='Customername')]
data_naren15=data_naren[(data_naren['Organization']=='Customername')]
data_naren16=data_naren[(data_naren['Organization']=='Customername')]
data_naren17=data_naren[(data_naren['Organization']=='Customername')]
data_naren18=data_naren[(data_naren['Organization']=='Customername')]
export_csv = data_naren1.to_csv (r'Customer1.csv', index = None, header=True)
export_csv = data_naren2.to_csv (r'Customer2.csv', index = None, header=True)
export_csv = data_naren3.to_csv (r'Customer3', index = None, header=True)
export_csv = data_naren4.to_csv (r'Customer4.csv', index = None, header=True)
export_csv = data_naren5.to_csv (r'Customer5.csv', index = None, header=True)
export_csv = data_naren6.to_csv (r'Customer6.csv', index = None, header=True)
export_csv = data_naren7.to_csv (r'Customer7.csv', index = None, header=True)
export_csv = data_naren8.to_csv (r'Customer8.csv', index = None, header=True)
export_csv = data_naren9.to_csv (r'Customer9.csv', index = None, header=True)
export_csv = data_naren10.to_csv (r'Customer10.csv', index = None, header=True)
export_csv = data_naren11.to_csv (r'Customer11.csv', index = None, header=True)
export_csv = data_naren12.to_csv (r'Customer12.csv', index = None, header=True)
export_csv = data_naren13.to_csv (r'Customer13.csv', index = None, header=True)
export_csv = data_naren14.to_csv (r'Customer14.csv', index = None, header=True)
export_csv = data_naren15.to_csv (r'Customer15.csv', index = None, header=True)
export_csv = data_naren16.to_csv (r'Customer16.csv', index = None, header=True)
export_csv = data_naren17.to_csv (r'Customer17.csv', index = None, header=True)
export_csv = data_naren18.to_csv (r'Customer18.csv', index = None, header=True)
```



☐ Name	Date modified	Type	Size
Customer1	21-12-2019 14:50	Microsoft Excel Com...	9 KB
Customer2	21-12-2019 14:50	Microsoft Excel Com...	11 KB
Customer3	21-12-2019 14:50	File	1 KB
Customer4	21-12-2019 14:50	Microsoft Excel Com...	2 KB
Customer5	21-12-2019 14:50	Microsoft Excel Com...	1 KB
Customer6	21-12-2019 14:50	Microsoft Excel Com...	3 KB
Customer7	21-12-2019 14:50	Microsoft Excel Com...	3 KB
Customer8	21-12-2019 14:50	Microsoft Excel Com...	4 KB
Customer9	21-12-2019 14:50	Microsoft Excel Com...	4 KB
Customer10	21-12-2019 14:50	Microsoft Excel Com...	2 KB
Customer11	21-12-2019 14:50	Microsoft Excel Com...	2 KB
Customer12	21-12-2019 14:50	Microsoft Excel Com...	7 KB
Customer13	21-12-2019 14:50	Microsoft Excel Com...	2 KB
Customer14	21-12-2019 14:50	Microsoft Excel Com...	10 KB
Customer15	21-12-2019 14:50	Microsoft Excel Com...	6 KB
Customer16	21-12-2019 14:50	Microsoft Excel Com...	1 KB
Customer17	21-12-2019 14:50	Microsoft Excel Com...	2 KB
Customer18	21-12-2019 14:50	Microsoft Excel Com...	3 KB

Source Code:

```
from tkinter import *
from tkinter import messagebox
import re,pymysql
from tkinter import Tk
from PIL import ImageTk, Image
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates
from seaborn import regplot
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from openpyxl import *
from tkinter import *
from tkinter import messagebox
import re , pymysql
from tkinter import Tk
from PIL import ImageTk, Image
from matplotlib.pyplot import title
from matplotlib.pyplot import subplots
from matplotlib.pyplot import subplot
from numpy import array
from numpy import mean
from numpy import nan
from pandas import read_csv
from pandas import to_numeric
from pandas import notnull
from fractions import Fraction
from matplotlib.pyplot import bar
from matplotlib.pyplot import ylim
from matplotlib.pyplot import xlabel
from matplotlib.pyplot import ylabel
from matplotlib.pyplot import title
from matplotlib.pyplot import xticks
from matplotlib.pyplot import subplots
from matplotlib.pyplot import subplot
from matplotlib.pyplot import show
from matplotlib.pyplot import scatter
from matplotlib.pyplot import plot
from matplotlib.pyplot import tight_layout
from seaborn import barplot
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import BOTH
from tkinter import BOTTOM
from tkinter import TOP
from tkinter import Text
from tkinter import END
from tkinter import Label
from tkinter import Scrollbar
from tkinter import messagebox
from tkinter import W
from tkinter import Entry
from csv import writer
from PIL import ImageTk, Image
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot
import numpy as np

data_naren=pd.read_csv("chiller.csv")
data_naren.dropna(axis=1)
```

```
# This function is used for adjusting window size and making the necessary
configuration on start of window
def adjustWindow(window):
    w = 800  # width for the window size
    h = 600  # height for the window size
    ws = screen.winfo_screenwidth()  # width of the screen
    hs = screen.winfo_screenheight()  # height of the screen
    x = (ws/2) - (w/2)  # calculate x and y coordinates for the Tk window
    y = (hs/2) - (h/2)
    window.geometry('%dx%d+%d+%d' % (w, h, x, y))  # set the dimensions of the
screen and where it is placed
    window.resizable(False, False)    # disabling the resize option for the window
    window.configure(background='white')    # making the background white of the
window

def adjustWindow3(window):
    w = 870  # width for the window size
    h = 600  # height for the window size
    ws = screen.winfo_screenwidth()  # width of the screen
    hs = screen.winfo_screenheight()  # height of the screen
    x = (ws/2) - (w/2)  # calculate x and y coordinates for the Tk window
    y = (hs/2) - (h/2)
    window.geometry('%dx%d+%d+%d' % (w, h, x, y))  # set the dimensions of the
screen and where it is placed
    window.resizable(False, False)    # disabling the resize option for the window
    window.configure(background='white')    # making the background white of the
window

# This function is used for adjusting window size and making the necessary
configuration on start of window
def adjustWindow1(window):
    w = screen.winfo_screenwidth()  # width of the screen
    h = screen.winfo_screenheight()  # height of the screen
    x = (w/2)  # calculate x and y coordinates for the Tk window
    y = (h/2)
    window.geometry('%dx%d+%d+%d' % (w, h, x, y))  # set the dimensions of the
screen and where it is placed
    #window.resizable(False, False)    # disabling the resize option for the window
    window.configure(background='white')    # making the background white of the
window

def adjustWindow2(window):
    w = 900
    h = 750
    ws = welcome_screen.winfo_screenwidth()
    hs = welcome_screen.winfo_screenheight()
    x = (ws/3.5) - (w/3.5)
    y = (hs/3.5) - (h/3.5)
    window.geometry('%dx%d+%d+%d' % (w, h, x, y))
    window.resizable(False, False)
# registration window
def register():
    global screen1, fullname, email, password, repassword, country, gender, tnc #
making all entry field variable global
    fullname = StringVar()
    email = StringVar()
    password = StringVar()
    repassword = StringVar()
    gender = StringVar()
    country = IntVar()
    tnc = IntVar()
    screen1 = Toplevel(screen)
    screen1.title("GPX DATA ANALYSIS")
    adjustWindow(screen1) # configuring the window
    Label(screen1, text="Registration Form", width='55', height="2", font=("Calibri", 22,
'bold'), fg='#d9660a').place(x=0, y=0)
    #Label(screen1, text="", bg='#174873', width='55', height='30').place(x=105, y=100)

    photo = PhotoImage(file="reg.png") # opening left side image - Note: If image is in
same folder then no need to mention the full path
    label = Label(screen1, image=photo, text="") # attaching image to the label
    label.place(x=-1, y=80)
    label.image = photo # it is necessary in Tkinter to keep a instance of image to display
image in label
```

```
    #Label(screen1, text="REGISTRATION FORM", font=("Open Sans", 22, 'bold'),
fg='white', bg='#174873', anchor=W).place(x=250, y=30)
    #Label(screen1, text="Registration Form", width='55', height="1", font=("Calibri", 22,
'bold'), fg='#d9660a').place(x=0, y=20)
    Label(screen1, text="Full Name:", font=("Open Sans", 11, 'bold'), fg='white',
bg='#174873', anchor=W).place(x=350, y=130)
    Entry(screen1, textvar=fullname).place(x=480, y=130)
    Label(screen1, text="Email ID:", font=("Open Sans", 11, 'bold'), fg='white',
bg='#174873', anchor=W).place(x=350, y=180)
    Entry(screen1, textvar=email).place(x=480, y=180)
    Label(screen1, text="Country:", font=("Open Sans", 11, 'bold'), fg='white',
bg='#174873', anchor=W).place(x=350, y=230)
    Radiobutton(screen1, text="India", variable=country, value=1,fg='white',
bg='#174873').place(x=480, y=230)
    Radiobutton(screen1, text="Other", variable=country, value=2,fg='white',
bg='#174873').place(x=550, y=230)

    Label(screen1, text="gender:", font=("Open Sans", 11, 'bold'), fg='white',
bg='#174873', anchor=W).place(x=350, y=280)
    list1 = ['Male','Female','Other']
    droplist = OptionMenu(screen1, gender, *list1)
    droplist.config(width=18)
    gender.set('--select your gender--')
    droplist.place(x=480, y=275)
    Label(screen1, text="Password:", font=("Open Sans", 11, 'bold'),
fg='white',bg='#174873', anchor=W).place(x=350, y=330)
    Entry(screen1, textvar=password, show="*").place(x=480, y=330)
    Label(screen1, text="Re-Password:", font=("Open Sans", 11, 'bold'),
fg='white',bg='#174873', anchor=W).place(x=350, y=380)
    entry_4 = Entry(screen1, textvar=repassword, show="*")
    entry_4.place(x=480, y=380)
    Checkbutton(screen1, text="I accept all terms and conditions", variable=tnc,
font=("Open Sans", 9, 'bold'), fg='brown').place(x=375, y=430)
    Button(screen1, text='Submit', width=15, font=("Open Sans", 13, 'bold'), bg='brown',
fg='black',command=register_user).place(x=330, y=480)
    Button(screen1, text='Back', width=10, font=("Open Sans", 13, 'bold'), bg='brown',
fg='black',command=screen1.destroy).place(x=550, y=480)
```

```
    photo = PhotoImage(file="log.png") # opening left side image - Note: If image is in
same folder then no need to mention the full path
    label = Label(screen1, image=photo, text="") # attaching image to the label
    label.place(x=130, y=0)
    label.image = photo # it is necessary in Tkinter to keep a instance of image to display
image in label



def register_user():
    if fullname.get() and email.get() and password.get() and repassword.get() and
country.get(): # checking for all empty values in entry field
        if gender.get() == "--select your gender--": # checking for selection of university
            Label(screen1, text="Please select your gender", fg="red",
                  font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)
            return
        else:
            if tnc.get(): # checking for acceptance of agreement
                if re.match("^.+@(\[?)[a-zA-Z0-9-.]+.([a-zA-Z]{2,3}|[0-9]{1,3})(]?)$",
email.get()): # validating the email
                    if password.get() == repassword.get(): # checking both password match or
not
                        # if u enter in this block everything is fine just enter the values in
database
                        country_value = 'India'
                        if country.get() == 2:

                            country_value = 'Other'
                        connection = pymysql.connect(host="localhost", user="root", passwd="",
database="bruteforce") # database connection
                        cursor = connection.cursor()
                        insert_query = "INSERT INTO registration_details (fullname, email,
password, country,gender) VALUES('"+ fullname.get() + "', '"+ email.get() + "', '"+
password.get() + "', '"+ country_value + "', '"+ gender.get() + "' );" # queries for inserting
values
                        cursor.execute(insert_query) # executing the queries
                        connection.commit() # commiting the connection then closing it.
                        connection.close() # closing the connection of the database
                        Label(screen1, text="Registration Sucess", fg="green", font=( "calibri",
11), width='30', anchor=W, bg='white').place(x=0, y=570) # printing successful
registration message
                        Button(screen1, text='Proceed to Login ->', width=20, font=("Open
Sans", 9, 'bold'), bg='brown', fg='white',command=screen1.destroy).place(x=170,
y=565) # button to navigate back to login page
                    else:
                        Label(screen1, text="Password does not match", fg="red", font=(
"calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)
                        return
                else:
                    Label(screen1, text="Please enter valid email id", fg="red", font=("calibri",
11), width='30', anchor=W, bg='white').place(x=0, y=570)
                    return
            else:
                Label(screen1, text="Please accept the agreement", fg="red",
                      font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)
                return
    else:
        Label(screen1, text="Please fill all the details", fg="red",
        font=("calibri", 11), width='30', anchor=W, bg='white').place(x=0, y=570)
        return

# login creditentials verification
def login_verify():
    global registrationID
    connection = pymysql.connect(host="localhost", user="root", passwd="",
database="bruteforce") # database connection
    cursor = connection.cursor()
    select_query =  "SELECT * FROM registration_details where email = '" +
username_verify.get() + "' AND password = '" + password_verify.get() + "';" # queries for
retrieving values
    cursor.execute(select_query) # executing the queries
    registration_info = cursor.fetchall()
    connection.commit() # commiting the connection then closing it.
    connection.close() # closing the connection of the database
    if registration_info:
        messagebox.showinfo("Congratulation", "Login Succesfull") # displaying message
for successful login
```

```
        registrationID = registration_info[0][0]
        welcome_page(registration_info) # opening welcome window
    else:
        messagebox.showerror("Error", "Invalid Username or Password") # displaying
message for invalid details


# welcome window
def welcome_page(registration_info):
    global screen2
    screen2 = Toplevel(screen)
    screen2.title("GPX DATA ANALYSIS")
    adjustWindow(screen2) # configuring the window
    Label(screen2, text="Welcome " + registration_info[0][1], width='47', height="2",
font=("Calibri", 25, 'bold'), fg='white', bg='#d9660a').place(x=0, y=0)
    Label(screen2, text="", bg='#174873', width='20', height='20').place(x=0, y=96)
    Message(screen2, text='“If we have data, let’s look at data. If all we have are
opinions, let’s go with mine.”\n\n - — Jim Barksdale', width='100', font=("Helvetica", 10,
'bold', 'italic'), fg='white', bg='#174873', anchor = CENTER).place(x=10, y=100)

    photo1 = PhotoImage(file="analy.png") # opening right side image - Note: If image is
in same folder then no need to mention the full path
    label1 = Label(screen2, image=photo1, text="") # attaching image to the label
    label1.place(x=150, y=96)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to
display image in label

    photo1 = PhotoImage(file="pay.png") # opening right side image - Note: If image is in
same folder then no need to mention the full path
    label1 = Label(screen2, image=photo1, text="") # attaching image to the label
    label1.place(x=180, y=0)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to
display image in label



    Button(screen2, text='Sales Prediction', width=30,height=2, font=("Open Sans", 13,
'bold'),command=mlwindow, bg='brown', fg='white').place(x=270, y=150)
    Button(screen2, text='MIS Generation', width=30,height=2, font=("Open Sans", 13,
'bold'),command=misgeneration, bg='brown', fg='white').place(x=270, y=210)
```

```
    Button(screen2, text='Chiller Data Analysis', width=30,height=2, font=("Open Sans",
13, 'bold'), bg='brown', fg='white',command=next_page).place(x=270, y=270)
    Button(screen2, text='Cross Connect Prediction', width=30,height=2, font=("Open
Sans", 13, 'bold'), bg='brown', fg='white',command=mlwindow2).place(x=270, y=330)
    Button(screen2, text='Compressorwise Analysis', width=30,height=2, font=("Open
Sans", 13, 'bold'), bg='brown', fg='white',command=companalysis).place(x=270, y=390)
    Button(screen2, text='Back', width=10, font=("Open Sans", 13, 'bold'), bg='brown',
fg='black',command=screen2.destroy).place(x=380, y=500)


def next_page():
    global screen3
    screen3 = Toplevel(screen)
    screen3.title("GPX DATA ANALYSIS")
    #adjustWindow(screen3)
    screen3.geometry("950x650")
    screen3.resizable(False,False)

    photo1 = PhotoImage(file="smok1.png") # opening right side image - Note: If image is
in same folder then no need to mention the full path
    label1 = Label(screen3, image=photo1, text="") # attaching image to the label
    label1.place(x=-5, y=50)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to
display image in label

    Label(screen3, text="--Functions-- ",width=40, height=1, font=("Open Sans", 30,
'bold'), bg='#174873', fg='white').place(x=0,y=0)

    Label(screen3, text="Page 1/3 ",width=8, height=1, font=("Open Sans", 13, 'bold'),
fg='black', bg='white').place(x=400,y=600)

    Button(screen3, text="1) Actual Capacity",width=100, height=2, bg='brown',
fg='white',font=("Open Sans", 10, 'bold'),command=question_1).place(x=80,y=84)
    Button(screen3, text="2) Entering Fluid Temperature",width=100, height=2,
bg='brown', fg='white',font=("Open Sans", 10,
'bold'),command=question_2).place(x=80,y=166)
    Button(screen3, text="3) Leaving Fluid Temperature",width=100,
height=2,font=("Open Sans", 10, 'bold'),command=question_3, bg='brown',
fg='white').place(x=80,y=248)
```

```
    Button(screen3, text="4) Condensation REF PRE",width=100, height=2,font=("Open Sans", 10, 'bold'),command=question_4, bg='brown', fg='white').place(x=80,y=332)
    Button(screen3, text="5) Differential Pressure",width=100, height=2,font=("Open Sans", 10, 'bold'),command=question_5, bg='brown', fg='white').place(x=80,y=416)
    Button(screen3, text="6) Condensation Saturation Temperature",width=100, height=2,font=("Open Sans", 10, 'bold'),command=question_6, bg='brown', fg='white').place(x=80,y=500)
    Button(screen3, text='< Back', width=10, font=("Open Sans", 13, 'bold'), bg='brown', fg='black',command=screen3.destroy).place(x=40, y=590)
    Button(screen3, text='Next >', width=10, font=("Open Sans", 13, 'bold'), bg='brown', fg='black',command=next_page1).place(x=800, y=590)


def next_page1():
    global screen311
    screen311 = Toplevel(screen)
    screen311.title("GPX DATA ANALYSIS")
    screen311.geometry("950x650")
    screen311.resizable(False,False)

    photo1 = PhotoImage(file="smok1.png") # opening right side image - Note: If image is in same folder then no need to mention the full path
    label1 = Label(screen311, image=photo1, text="") # attaching image to the label
    label1.place(x=-5, y=50)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to display image in label

    Label(screen311, text="--Functions-- ",width=40, height=1, font=("Open Sans", 30, 'bold'), bg='#174873', fg='white').place(x=0,y=0)

    Label(screen311, text="Page 2/3 ",width=8, height=1, font=("Open Sans", 13, 'bold'), fg='black', bg='white').place(x=400,y=600)


    Button(screen311, text="7) Evaporator Saturation Temperature",width=90, height=2,font=("Open Sans", 10, 'bold'),command=question_7, bg='brown', fg='white').place(x=80,y=84)
    Button(screen311, text="8) EXPV Position",width=95, height=2, bg='brown', fg='white',font=("Open Sans", 10, 'bold'),command=question_8).place(x=80,y=164)
    Button(screen311, text="9) Suction Temperature",width=95, height=2, bg='brown', fg='white',font=("Open Sans", 10, 'bold'),command=question_9).place(x=80,y=248)
    Button(screen311, text="10) Discharge Temperature",width=90, height=2,font=("Open Sans", 10, 'bold'),command=question_10, bg='brown', fg='white').place(x=80,y=332)
    Button(screen311, text="11) Suction Superheat",width=90, height=2,font=("Open Sans", 10, 'bold'),command=question_11, bg='brown', fg='white').place(x=80,y=416)
    Button(screen311, text="12) Discharge Superheat",width=90, height=2,font=("Open Sans", 10, 'bold'),command=question_12 ,bg='brown', fg='white').place(x=80,y=500)

    Button(screen311, text='< Back', width=10, font=("Open Sans", 13, 'bold'), bg='brown', fg='black',command=screen311.destroy).place(x=40, y=590)

    Button(screen311, text='Next >', width=10, font=("Open Sans", 13, 'bold'), bg='brown', fg='black',command=next_page2).place(x=800, y=590)

def next_page2():
    global screen312
    screen312 = Toplevel(screen)
    screen312.title("BRUTE FORCE")
    screen312.geometry("950x650")
    screen312.resizable(False,False)

    photo1 = PhotoImage(file="smok1.png") # opening right side image - Note: If image is in same folder then no need to mention the full path
    label1 = Label(screen312, image=photo1, text="") # attaching image to the label
    label1.place(x=-5, y=50)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to display image in label

    Label(screen312, text="--Functions-- ",width=40, height=1, font=("Open Sans", 30, 'bold'), bg='#174873', fg='white').place(x=0,y=0)

    Label(screen312, text="Page 3/3 ",width=8, height=1, font=("Open Sans", 13, 'bold'), fg='black', bg='white').place(x=400,y=600)
    Button(screen312, text="13) Condensation Ref Pressure",width=90,
height=2,font=("Open Sans", 10, 'bold'), bg='brown',
fg='white',command=question_13).place(x=80,y=84)
    Button(screen312, text="14) Evaporator Pressure",width=90, height=2,font=("Open
Sans", 10, 'bold'), bg='brown', fg='white',command=question_14).place(x=80,y=164)


    Button(screen312, text='< Back', width=10, font=("Open Sans", 13, 'bold'),
bg='brown', fg='black',command=screen312.destroy).place(x=40, y=590)

def main_screen():
    global screen, username_verify, password_verify
    screen = Tk()
    username_verify = StringVar()
    password_verify = StringVar()
    screen.title("DATA ANALYSIS")
    adjustWindow(screen)

    img2 = ImageTk.PhotoImage(Image.open("bigdata.png"))
    d1 = Label(image= img2)
    d1.place(x =-1,y = 78)

    Label(screen,text="GPX DATA
ANALYSIS",width="55",height="2",font=("Calibri",22,'bold'),bg='white',fg='#174873').pac
k()
    Label(screen, text="Please enter details below to login", bg='#174873',
fg='white').place(x=312,y=150)
    Label(screen, text="Username * ", font=("Open Sans", 10, 'bold'), bg='#174873',
fg='white').place(x=358,y=192)
    Entry(screen, textvar=username_verify).place(x=337,y=215)
    Label(screen, text="Password * ", font=("Open Sans", 10, 'bold'), bg='#174873',
fg='white').place(x=358,y=262)
    Entry(screen, textvar=password_verify, show="*").place(x=337,y=285)
    Button(screen, text="LOGIN", bg="#e79700", width=15, height=1, font=("Open Sans",
13, 'bold'), fg='white', command=login_verify).place(x=320,y=325)
    Button(screen, text="New User? Register Here", height="2", width="30",
bg='#e79700', font=("Open Sans", 10, 'bold'), fg='white',
command=register).place(x=270,y=380)
```

```
    img1 = ImageTk.PhotoImage(Image.open("bft.png"))
    c1 = Label(image= img1)
    c1.place(x =142,y = 0)



    screen.mainloop()

def question_1():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual
Capacity'], name="COND REF PRESS COMP1",line_color='green'))
    fig1.update_layout(title_text='Actual Capacity',xaxis_rangeslider_visible=True)


    fig1.show()
    plot(fig1)
def question_2():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid
temp'], name="COND REF PRESS COMP1",line_color='green'))
   fig1.update_layout(title_text='Entering Fluid
temperature',xaxis_rangeslider_visible=True)


   fig1.show()
   plot(fig1)

def question_3():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid
temp'], name="COND REF PRESS COMP1",line_color='green'))
   fig1.update_layout(title_text='Leaving Fluid
temperature',xaxis_rangeslider_visible=True)


   fig1.show()
   plot(fig1)
```

```
def question_14():
  fig1 = go.Figure()
  fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP1'], name="EVAPORATIVE PRESSURE COMPRESSOR 1",line_color='green'))
  fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP2'], name="EVAPORATIVE PRESSURE COMPRESSOR
1",line_color='deepskyblue'))
  fig1.update_layout(title_text='Evaporator Pressure
Compressorwise',xaxis_rangeslider_visible=True)


  fig1.show()
  plot(fig1)

def question_13():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF
PRESS COMP1'], name="COND REF PRESS COMP1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF
PRESS COMP2'], name="COND REF PRESS COMP2",line_color='green'))

   fig1.update_layout(title_text='Condensation Ref Pressure of
Compressors',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)

def question_4():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure
COMP1'], name="Oil Pressure COMP 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure
COMP2'], name="Oil Pressure COMP2",line_color='green'))

   fig1.update_layout(title_text='Oil Pressure of
Compressors',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)
```

```
def question_5():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'],
name="DP",line_color='deepskyblue'))

   fig1.update_layout(title_text='Differential Pressure',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)

def question_6():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT
TEMP COMP 1'], name="COND SAT TEMP COMP 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT
TEMP COMP 2'], name="COND SAT TEMP COMP 2",line_color='green'))

   fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)

def question_7():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT
TEMP COM1'], name="EVAP SAT TEMP COMP 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT
TEMP COM2'], name="EVAP SAT TEMP COMP 2",line_color='green'))

   fig1.update_layout(title_text='Evaporator Saturation
Temperature',xaxis_rangeslider_visible=True)
   fig1.show()
   plot(fig1)

def question_8():
   fig1 = go.Figure()
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position
COM 1'], name="EXPV Position COM 1",line_color='deepskyblue'))
   fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position
COM 2'], name="EXPV Position COM 2",line_color='green'))
```

```
    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

def question_9():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION TEMP COMP 1'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION TEMP COMP 2'], name="SUCTION TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

def question_10():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE TEMP COMP 1'], name="DISCHARGE TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE TEMP COMP 2'], name="DISCHARGE TEMP COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

def question_11():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH COMP 1'], name="SUCTION SH COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH COMP 2'], name="SUCTION SH COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

def question_12():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH SH COMP 1'], name="DISCH SH COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH SH COMP 2'], name="DISCH SH COMP 2",line_color='green'))

    fig1.update_layout(title_text='DP',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)

#sales prediction
def mlwindow():
    global screen55
    screen55 = Tk()
    screen55.title('Question-1')
    adjustWindow1(screen55)

    Label(screen55, text='The Prediction for sales is as follows' ,font=('helvetica', 10, 'bold')).pack()
    Label(screen55, text='To predict for a particular year click the button below' ,font=('helvetica', 10, 'bold')).pack()
    Button(screen55,text='Get the Prediction',command=mlpredict, bg='brown', fg='white', font=('helvetica', 9, 'bold')).pack()
    fig, ax = plt.subplots()

    data=np.array([
    [2017,119],
    [2018,75],
    [2019,150]])

    x,y=data.T
    plt.ylabel("Number of Clients")
    plt.xlabel("Year")
    plt.scatter(x,y)


    data1=np.array([
    [2017,118.99],
    [2018,114.66],
    [2019,110.33]])
    x,y=data1.T
    plt.ylabel("Number of Clients")
    plt.xlabel("Year")
    plt.scatter(x,y)
    plt.plot(x,y)



    canvas = FigureCanvasTkAgg(fig, screen55)
    canvas.draw()
    canvas.get_tk_widget().pack(side = TOP, fill = BOTH, expand = True)

    button = Button(screen55, text="Quit", command=screen55.destroy)
    button.pack(side=BOTTOM)

def mlpredict():
    import tkinter as tk

    root= tk.Tk()

    canvas1 = tk.Canvas(root, width = 400, height = 300,  relief = 'raised')
    canvas1.pack()

    label1 = tk.Label(root, text='Prediction of Number of New Clients')
    label1.config(font=('helvetica', 14))
    canvas1.create_window(200, 25, window=label1)

    label2 = tk.Label(root, text='Type your Number:')
    label2.config(font=('helvetica', 10))
    canvas1.create_window(200, 100, window=label2)

    entry1 = tk.Entry (root)
    canvas1.create_window(200, 140, window=entry1)

    def getSquareRoot ():

        x1 = entry1.get()
        x2=(float(x1)*(-4.33))+(8852.6)
        x3=str(x2)
        label3 = tk.Label(root, text= 'The Prediction of Sales" is:',font=('helvetica', 10))
        canvas1.create_window(200, 210, window=label3)

        label4 = tk.Label(root, text=x3 ,font=('helvetica', 10, 'bold'))
        canvas1.create_window(200, 230, window=label4)

    button1 = tk.Button(root,text='Get the Prediction', command=getSquareRoot,
bg='brown', fg='white', font=('helvetica', 9, 'bold'))

    canvas1.create_window(200, 180, window=button1)

    root.mainloop()

#cross connect prediction
def mlwindow2():
    global screen65
    screen65 = Tk()
    screen65.title('Question-1')
    adjustWindow1(screen65)

    Label(screen65, text='The Prediction for Cross Connects is as follows'
,font=('helvetica', 10, 'bold')).pack()
    Label(screen65, text='To predict for a particular year click the button below'
,font=('helvetica', 10, 'bold')).pack()
    Button(screen65,text='Get the Prediction',command=mlpredict2, bg='brown',
fg='white', font=('helvetica', 9, 'bold')).pack()
    fig, ax = plt.subplots()

    data=np.array([
    [2016,258],
    [2017,569],
    [2018,696],
    [2019,719]])

    x,y=data.T
    plt.ylabel("Number of Cross Connects")
    plt.xlabel("Year")
    plt.scatter(x,y)
    data1=np.array([
    [2016,330.175],
    [2017,483.725],
    [2018,637.275],
    [2019,790.825]])

    x,y=data1.T
    plt.ylabel("Number of Cross Connects")
    plt.xlabel("Year")
    plt.scatter(x,y)
    plt.plot(x,y)



    canvas = FigureCanvasTkAgg(fig, screen65)
    canvas.draw()
    canvas.get_tk_widget().pack(side = TOP, fill = BOTH, expand = True)

    button = Button(screen65, text="Quit", command=screen55.destroy)
    button.pack(side=BOTTOM)

def mlpredict2():
    import tkinter as tk

    root1= tk.Tk()

    canvas1 = tk.Canvas(root1, width = 400, height = 300,  relief = 'raised')
    canvas1.pack()

    label1 = tk.Label(root1, text='Prediction of Cross Connects')
    label1.config(font=('helvetica', 14))
    canvas1.create_window(200, 25, window=label1)

    label2 = tk.Label(root1, text='Type your Number:')
    label2.config(font=('helvetica', 10))
    canvas1.create_window(200, 100, window=label2)
    entry1 = tk.Entry (root1)
    canvas1.create_window(200, 140, window=entry1)

    def getSquareRoot2 ():

        x1 = entry1.get()
        x2=(float(x1)*153.55)-(309226.625)
        x3=str(x2)
        label3 = tk.Label(root1, text= 'The Prediction of Cross Connects"
is:',font=('helvetica', 10))
        canvas1.create_window(200, 210, window=label3)

        label4 = tk.Label(root1, text=x3 ,font=('helvetica', 10, 'bold'))
        canvas1.create_window(200, 230, window=label4)

    button1 = tk.Button(root1,text='Get the Prediction', command=getSquareRoot2,
bg='brown', fg='white', font=('helvetica', 9, 'bold'))

    canvas1.create_window(200, 180, window=button1)

    root1.mainloop()

def comppage():
    global screen369
    screen369 = Toplevel(screen)
    screen369.title("GPX DATA ANALYSIS")
    screen369.geometry("950x650")
    screen369.resizable(False,False)

    photo1 = PhotoImage(file="smok1.png") # opening right side image - Note: If image is
in same folder then no need to mention the full path
    label1 = Label(screen369, image=photo1, text="") # attaching image to the label
    label1.place(x=-5, y=50)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to
display image in label

    Label(screen369, text="--Functions-- ",width=40, height=1, font=("Open Sans", 30,
'bold'), bg='#174873', fg='white').place(x=0,y=0)
```

```
    Label(screen369, text="Page 2/3 ",width=8, height=1, font=("Open Sans", 13, 'bold'), fg='black', bg='white').place(x=400,y=600)


    Button(screen369, text="1) Compressor 1",width=90, height=2,font=("Open Sans", 10, 'bold'),command=question_7, bg='brown', fg='white').place(x=80,y=84)
    Button(screen369, text="8) Compressor 2",width=95, height=2, bg='brown', fg='white',font=("Open Sans", 10, 'bold'),command=question_8).place(x=80,y=164)


    Button(screen369, text='< Back', width=10, font=("Open Sans", 13, 'bold'), bg='brown', fg='black',command=screen369.destroy).place(x=40, y=590)

def comp1():

    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual Capacity'], name="Actual Capacity",line_color='red'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid temp'], name="Entering Fluid Temperature",line_color='blue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid temp'], name="Leaving Fluid Temperature",line_color='green'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'], name="Differential Pressure",line_color='yellow'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION TEMP COMP 1'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF PRESS COMP1'], name="Condensation Ref Pressure ",line_color='orange'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure COMP1'], name="Oil Pressure",line_color='pink'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT TEMP COMP 1'], name="Condensation SatTemperature",line_color='brown'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT TEMP COM1'], name="Evaporator Sat Temperature",line_color='purple'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position COM 1'], name="SUCTION TEMP COMP 2",line_color='magenta'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS COMP1'], name="Evaporator Pressure",line_color='violet'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS COMP1'], name="SUCTION TEMP COMP 2",line_color='black'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE TEMP COMP 1'], name="Discharge Temperature",line_color='yellowgreen'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH COMP 1'], name="Suction superheat",line_color='lavender'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH     SH COMP 1'], name="Discharge Superheat",line_color='cyan'))

    fig1.update_layout(title_text='COMPRESSOR 1',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)


def comp2():
    fig1 = go.Figure()
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Actual Capacity'], name="Actual Capacity",line_color='red'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Entering Fluid temp'], name="Entering Fluid Temperature",line_color='blue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['Leaving Fluid temp'], name="Leaving Fluid Temperature",line_color='green'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DP'], name="Differential Pressure",line_color='yellow'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION TEMP COMP 2'], name="SUCTION TEMP COMP 1",line_color='deepskyblue'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND REF PRESS COMP2'], name="Condensation Ref Pressure ",line_color='orange'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['OIL Pressure COMP2'], name="Oil Pressure",line_color='pink'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['COND SAT TEMP COMP 2'], name="Condensation SatTemperature",line_color='brown'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP SAT TEMP COM2'], name="Evaporator Sat Temperature",line_color='purple'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EXPV Position COM 2'], name="SUCTION TEMP COMP 2",line_color='magenta'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS COMP2'], name="Evaporator Pressure",line_color='violet'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['EVAP PRESS
COMP2'], name="SUCTION TEMP COMP 2",line_color='black'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCHARGE
TEMP COMP 2'], name="Discharge Temperature",line_color='yellowgreen'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['SUCTION SH
COMP 2'], name="Suction superheat",line_color='lavender'))
    fig1.add_trace(go.Scatter(x=data_naren['Time Snap'], y=data_naren['DISCH    SH
COMP 2'], name="Discharge Superheat",line_color='cyan'))

    fig1.update_layout(title_text='COMPRESSOR 2',xaxis_rangeslider_visible=True)
    fig1.show()
    plot(fig1)



def misgeneration():
    data_naren=pd.read_csv("sunilsir.csv",usecols=[0,1,7,8,11,12,13,14])

data_naren1=data_naren[(data_naren['Organization']=='Customername')]
data_naren2=data_naren[(data_naren['Organization']=='Customername')]
data_naren3=data_naren[(data_naren['Organization']=='Customername')]
data_naren4=data_naren[(data_naren['Organization']=='Customername')]
data_naren5=data_naren[(data_naren['Organization']=='Customername')]
data_naren6=data_naren[(data_naren['Organization']=='Customername')]
data_naren7=data_naren[(data_naren['Organization']=='Customername')]
data_naren8=data_naren[(data_naren['Organization']=='Customername')]
data_naren9=data_naren[(data_naren['Organization']=='Customername')]
data_naren10=data_naren[(data_naren['Organization']=='Customername')]
data_naren11=data_naren[(data_naren['Organization']=='Customername')]
data_naren12=data_naren[(data_naren['Organization']=='Customername')]
data_naren13=data_naren[(data_naren['Organization']=='Customername')]
data_naren14=data_naren[(data_naren['Organization']=='Customername')]
data_naren15=data_naren[(data_naren['Organization']=='Customername')]
data_naren16=data_naren[(data_naren['Organization']=='Customername')]
data_naren17=data_naren[(data_naren['Organization']=='Customername')]
data_naren18=data_naren[(data_naren['Organization']=='Customername')]


export_csv = data_naren1.to_csv (r'Customer1.csv', index = None, header=True)
export_csv = data_naren2.to_csv (r'Customer2.csv', index = None, header=True)
export_csv = data_naren3.to_csv (r'Customer3', index = None, header=True)
export_csv = data_naren4.to_csv (r'Customer4.csv', index = None, header=True)
export_csv = data_naren5.to_csv (r'Customer5.csv', index = None, header=True)
export_csv = data_naren6.to_csv (r'Customer6.csv', index = None, header=True)
export_csv = data_naren7.to_csv (r'Customer7.csv', index = None, header=True)
export_csv = data_naren8.to_csv (r'Customer8.csv', index = None, header=True)
export_csv = data_naren9.to_csv (r'Customer9.csv', index = None, header=True)
export_csv = data_naren10.to_csv (r'Customer10.csv', index = None, header=True)
export_csv = data_naren11.to_csv (r'Customer11.csv', index = None, header=True)
export_csv = data_naren12.to_csv (r'Customer12.csv', index = None, header=True)
export_csv = data_naren13.to_csv (r'Customer13.csv', index = None, header=True)
export_csv = data_naren14.to_csv (r'Customer14.csv', index = None, header=True)
export_csv = data_naren15.to_csv (r'Customer15.csv', index = None, header=True)
export_csv = data_naren16.to_csv (r'Customer16.csv', index = None, header=True)
export_csv = data_naren17.to_csv (r'Customer17.csv', index = None, header=True)
export_csv = data_naren18.to_csv (r'Customer18.csv', index = None, header=True)


def companalysis():
    global screen333
    screen333 = Toplevel(screen)
    screen333.title("GPX DATA ANALYSIS")
    screen333.geometry("950x650")
    screen333.resizable(False,False)

    photo1 = PhotoImage(file="smok1.png") # opening right side image - Note: If image is
in same folder then no need to mention the full path
    label1 = Label(screen333, image=photo1, text="") # attaching image to the label
    label1.place(x=-5, y=50)
    label1.image = photo1 # it is necessary in Tkinter to keep a instance of image to
display image in label

    Label(screen333, text="--Functions-- ",width=40, height=1, font=("Open Sans", 30,
'bold'), bg='#174873', fg='white').place(x=0,y=0)



    Button(screen333, text="1) Compressor 1",width=90, height=2,font=("Open Sans",
10, 'bold'),command=comp1, bg='brown', fg='white').place(x=80,y=84)
```

```
    Button(screen333, text="2) Compressor 2",width=95, height=2, bg='brown',
fg='white',font=("Open Sans", 10, 'bold'),command=comp2).place(x=80,y=164)

    Button(screen333, text='< Back', width=10, font=("Open Sans", 13, 'bold'),
bg='brown', fg='black',command=screen333.destroy).place(x=40, y=590)
main_screen()
```

OCR(Optical character recognition) Code:

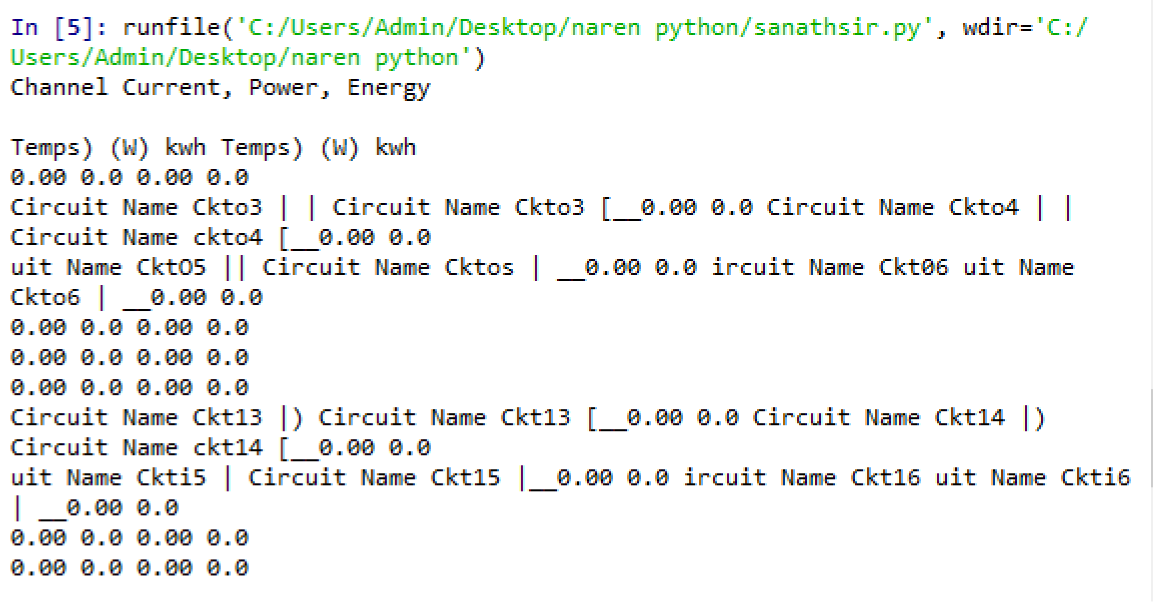
This OCR (Optical Character Recognition) code converts the character and number present in an image to text format within seconds.This overcomes the problem of the NOC members to manually write the data that was restricted to be copied from a webpage.
For quite some time the NOC members were facing this problem of manually entering the data from screenshots of the webpage taken. This not only increased the time of doing a particular task but also hampered the priority to be given to a particular task that required attention at any particular given time.

The following code has been written on the basis of a code that already exists on GitHub

Source Code

```
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program
Files\Tesseract-OCR\tesseract.exe'

try:
    from PIL import Image
except ImportError:
    import Image
def ocr_core(filename):
    """
    This function will handle the core OCR processing of images.
    """
    text = pytesseract.image_to_string(Image.open(filename))  # We'll use Pillow's Image
class to open the image and pytesseract to detect the string in the image
    return text

print(ocr_core('sanathsir.png'))
```

Output



```
In [5]: runfile('C:/Users/Admin/Desktop/naren python/sanathsir.py', wdir='C:/
Users/Admin/Desktop/naren python')
Channel Current, Power, Energy

Temps) (W) kwh Temps) (W) kwh
0.00 0.0 0.00 0.0
Circuit Name Ckto3 | | Circuit Name Ckto3 [__0.00 0.0 Circuit Name Ckto4 | |
Circuit Name ckto4 [__0.00 0.0
uit Name CktO5 || Circuit Name Cktos | __0.00 0.0 ircuit Name Ckt06 uit Name
Ckto6 | __0.00 0.0
0.00 0.0 0.00 0.0
0.00 0.0 0.00 0.0
0.00 0.0 0.00 0.0
Circuit Name Ckt13 |) Circuit Name Ckt13 [__0.00 0.0 Circuit Name Ckt14 |)
Circuit Name ckt14 [__0.00 0.0
uit Name Ckti5 | Circuit Name Ckt15 |__0.00 0.0 ircuit Name Ckt16 uit Name Ckti6
| __0.00 0.0
0.00 0.0 0.00 0.0
0.00 0.0 0.00 0.0
```

Customized Calculation program

Mr Yashwant Kulkarni wanted a customized program to calculate the total power consumed by the facility in kVA units which required filtering and sorting of data as per specific device names and getting the maximum value of each device parameter. These maximum values were then added and a subtraction operation was performed between two values. One being the addition of the maximum total kVA and the other being the addition of kVA consumed as per a specific device.

The Source Code and the output is given below:

Sorting data as per device

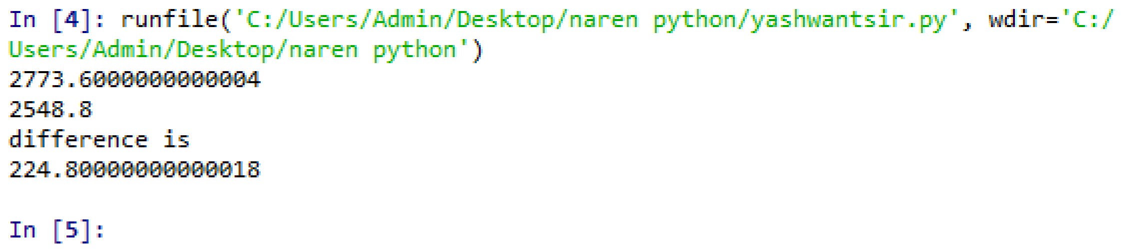
```
import pandas as pd

data_naren=pd.read_csv("load.csv")
data_naren['Total']= data_naren.iloc[:, 4:8].sum(axis=1)

data_naren1=data_naren[(data_naren['Device No.']=='SH000198')]
data_naren2=data_naren[(data_naren['Device No.']=='SH000269')]
data_naren3=data_naren[(data_naren['Device No.']=='SH000458')]
data_naren4=data_naren[(data_naren['Device No.']=='SH000492')]

export_csv = data_naren1.to_csv (r'Device1.csv', index = None, header=True)
export_csv = data_naren2.to_csv (r'Device2.csv', index = None, header=True)
export_csv = data_naren3.to_csv (r'Device3.csv', index = None, header=True)
export_csv = data_naren4.to_csv (r'Device4.csv', index = None, header=True)
```

The calculations

```
import pandas as pd

data_naren=pd.read_csv("load.csv")
data_naren['Total']= data_naren.iloc[:, 4:8].sum(axis=1)

data_naren1=pd.read_csv("Device1.csv")
data_naren2=pd.read_csv("Device2.csv")
```

```
data_naren3=pd.read_csv("Device3.csv")
data_naren4=pd.read_csv("Device4.csv")

x1=data_naren1['kVA'].max()
x2=data_naren2['kVA'].max()
x3=data_naren3['kVA'].max()
x4=data_naren4['kVA'].max()

b=x1+x2+x3+x4
print(x1+x2+x3+x4)

data_naren['Total kVa']=data_naren1['kVA']+data_naren2['kVA']+data_naren3['kVA']+data_naren4['kVA']
print(data_naren['Total kVa'].max())
a=data_naren['Total kVa'].max()
print('difference is')
print(b-a)
```

Output



```
In [4]: runfile('C:/Users/Admin/Desktop/naren python/yashwantsir.py', wdir='C:/
Users/Admin/Desktop/naren python')
2773.6000000000004
2548.8
difference is
224.80000000000018

In [5]:
```