INTERNSHIP REPORT– KALGUDI DIGITAL PVT LIMITED

A Project Report submitted in partial fulfillment of the requirements for the awardof the degree of

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

KADIYALA P B ROHIT BHARADWAJ, 121910313006

Under the esteemed guidance of

Mr. Sreekanth Voleti

(Head of the core)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

May-June 2022

DECLARATION

I, hereby declare that the Internship report is an original work done in the **"KALGUDI DIGITAL PVT LIMITED"** submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:07-11-2022

121910313006 KADIYALA P B ROHIT BHARADWAJ Signature: 

CERTIFICATE

Kalgudi

24-Jun-2022

To whom so ever it may concern

This is to certify that Mr. Kadiyala P B Rohit Bharadwaj of Gandhi Institute of Technology and Management has done his project in Building Knowledge Graphs and Graph Databases for FPOs (Using Neo4j and Python) Yield and Profit Prediction Tool for FPOs (Tkinter and Python Data Science) from 16 May 2022 to 24 June 2022 as a part of fulfillment of his curriculum in B Tech-CSE under the guidance of Mr Sreekanth Voleti,(Head of Core).

We take this opportunity of wishing him the very best in his future endeavors.

Thanking you,

For **Kalgudi Digital Pvt. Ltd.**

Human Resources

TABLE OF CONTENTS

1. ABSTRACT

The contribution of agriculture in GDP has declining recent past from 50 per cent in 1950 to 16.5 per cent in 2019-20. In current Covid-19 pandemic, this is the only sector which is contributed positively with 3.4 per cent growth in both the first quarter of the financial year 2020-21. Farmers face different challenges such as scarcity of land and water sources, impassable roads, unavailability of better financial services and new technologies. Government wants to double the farmers' income in 2022-23 by addressing the challenges of farm sector. The government affirmed that farmer producer organizations (FPO) are the most appropriate institutional form around which farmers can mobilize and build their capacity to collectively leverage their production and marketing strength. The members of FPOs are smallholder farmers who organize themselves with the objective of improving farm income through improved production, marketing, and local processing activities. unfortunately, majority of the members in FPOs lack of knowledge on farm inputs, cost of cultivation and profitability. Hence we developed an optimization tool which helps the farmers in planning their farming by allocating most profitable crops and develop an farm by allocating different crops which can result in more returns To make work easy for the FPOs to analyze their farm and farm inputs, the current tools gathers information on climate forecast from India Meteorological Department, yield from crop simulation models and prices from price forecasts developed by ANGRAU. Using all this information the tools provide optimum profitable cropping pattern to the FPO farmers to maximize their profits

2. INTRODUCTION

Collectivization of producers, especially small and marginal farmers, into producer organizations has emerged as one of the most effective pathways to address the many challenges of agriculture, most importantly, improved access to technology, inputs and markets. The Department of Agriculture and Cooperation, Ministry of Agriculture, Government of India has identified farmer producer organizations to be registered under the special provisions of the Companies Act, 2013 as the most appropriate institutional form to mobilize farmers and build their capacity to collectively leverage their production and marketing strength. Though several studies have so far reviewed performances of FPOs in India, most of them evaluated the status of farmer based institutions like cooperatives, mutually aided cooperative credit societies, associations under NGOs etc. Indeed, there is a paucity of literature and studies on FPOs. Also, different implementation models are being adopted by different FPOs operational in various states. There is a lot of problem for FPOs in identifying correct fertilizer composition. In this process we planned developed a tool to calculate fertilizer requirements, yield and profit maximization options for the FPO farmers. This tool gathers values from dataset of University fertilizer recommendations and calculates the correct composition of each component of fertilizer so that there will be no excess or lesser usage of fertilizer doses. Furthermore, it also predicts the Yield and profit for that particular crop in particular area based on IMD (India Meteorological Department) seasonal climate forecast using crop simulation models. This tools helps the FPOs and the registered farmers under such FPO to maximize their profit. Also, A Knowledge graphs are built in the Neo4j software in order to represent entities like Farmer, FPOs, Crops, Stores, Fertilizers, etc.

3. LITERATURE REVIEW

Agricultural extension systems are stretched beyond physical limits while assisting farmers during crop production in the developing world. Information and Communication Technologies are booming worldwide with the introduction of mobile applications in smart phones. Hence several people have come up with various tools and apps to assist the farmers in gathering latest research results and timely contextual information. In this literature review we tried to gather various latest tools developed different startup companies, organizations to help FPOs and farming community.

Indev consultancy (2022) developed an automation software for the Farmer Producer Organization (FPO) which helps small and poor farmers of India to get organized. The application framework will help farmers to participate in high-value markets to sell their products. Further to train the farmers and track their growth, they have developed a web-based dashboard and a Mobile Application with monitoring and evaluation systems. It will provide training, advisory, and support to the farmers and also gather the relevant information to help them compete in the markets.

Kehti Buddy (2022) has developed a platform to make the entire farming process easier for the FPO members. The members will get starting from sowing advice to following climate-resilient farming practices, farmers can get all necessary information via a farm app. FPOs can also view weather forecasts and other information about farming practices, request for input materials or even update data on the app if needed

FarmERP (2022) is an intelligent and next-generation farm management platform. FarmERP's agriculture management system can help FPOs and FPCs manage and optimize various business operations across all plots. FarmERP can empower them with next-generation farm management technology for ensuring high productivity across all their farmer partners' farms.

ICRISAT and its partners have developed an advisory system for FPO registered farmers called "Intelligent agricultural Systems Advisory Tool – ISAT" capable of generating and disseminating data driven location specific advisories that assist farmers in anticipating and responding to the emerging conditions through the season. Using a decision tree approach, a structured and systematic approach to decision making was devised that considers the insights obtained from the analysis of historical climatic conditions, climate and weather forecasts and prevailing environmental conditions (Rao et al., 2019).

Progressive Environment Agriculture Technologies (PEAT), a German startup company developed a mobile application 'Plantix' that assists in detecting damage on plants with the help of a smart phone image for the benefit farmers and FPO members. Mobile internet connectivity enables uploading of image (time and geo-tagged) to a cloud based server where machine learning based algorithms are used to process the optical patterns and detect the diseases. This information is conveyed back to the user within few seconds along with suggested knowledge on disease management including biological and chemical control methods (Srikanth and Alexaander 2018)

4. PROBLEM IDENTIFICATION & OBJECTIVES

Majority of farmers doesn't have the basic knowledge about the proper recommended fertilizer quantities and other inputs they should use for each crop they are growing as majority of them are illiterates. So they will elect a FPO president among them who have more knowledge on agriculture. We designed a Fertilizer tool and Yield and profit prediction tool which helps the FPOs to train the farmers on cropping pattern they have to follow based on seasonal climate forecast and also how to optimize their resources such as land, fertilizers etc., in order to maximize their profits by careful allocation of land with various crops.

5. SYSTEM METHODOLOGY

DATA PREPROCESSING

We performed the data preprocessing on the required data to reduce the feature [i.e columns]. checked for null values, duplicate rows, in-consistence of data and removed those using Pandas

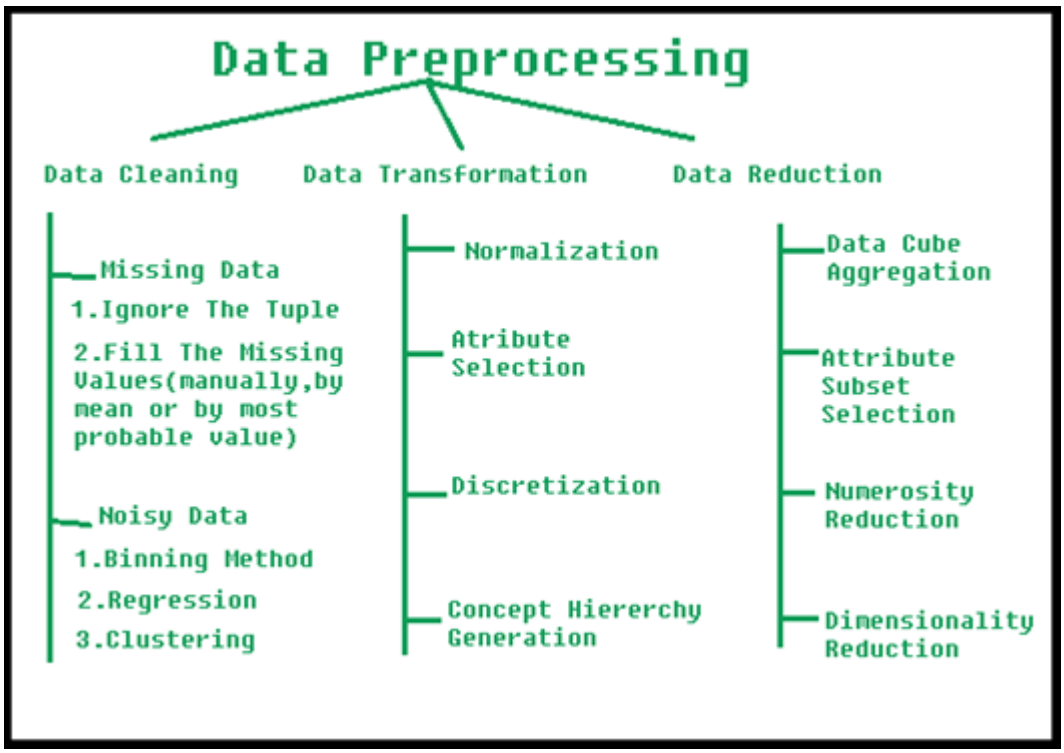


Fig 1. Diagram showing details of data preprocessing

6. OVERVIEW OF TECHNOLOGIES

6.1. TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. We used Tkinter library to design our front end of our tool. We used this package to make our tool more interactive, usewer friendly which provides the user great satisfaction.

6.2. MATPLOTLIB

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. it helps in creating and publication of quality plots. It also helps in making interactive figures that can zoom, pan, and update. We used this library to visualize bar graphs for comparing cost of cultivation and returns for a specific crop sown in a particular area.

6.3. PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” and “labeled” data both easy and intuitive. We used this package to read the CSV files and use them as data frames for further calculations. We used this packages mainly for data preprocessing

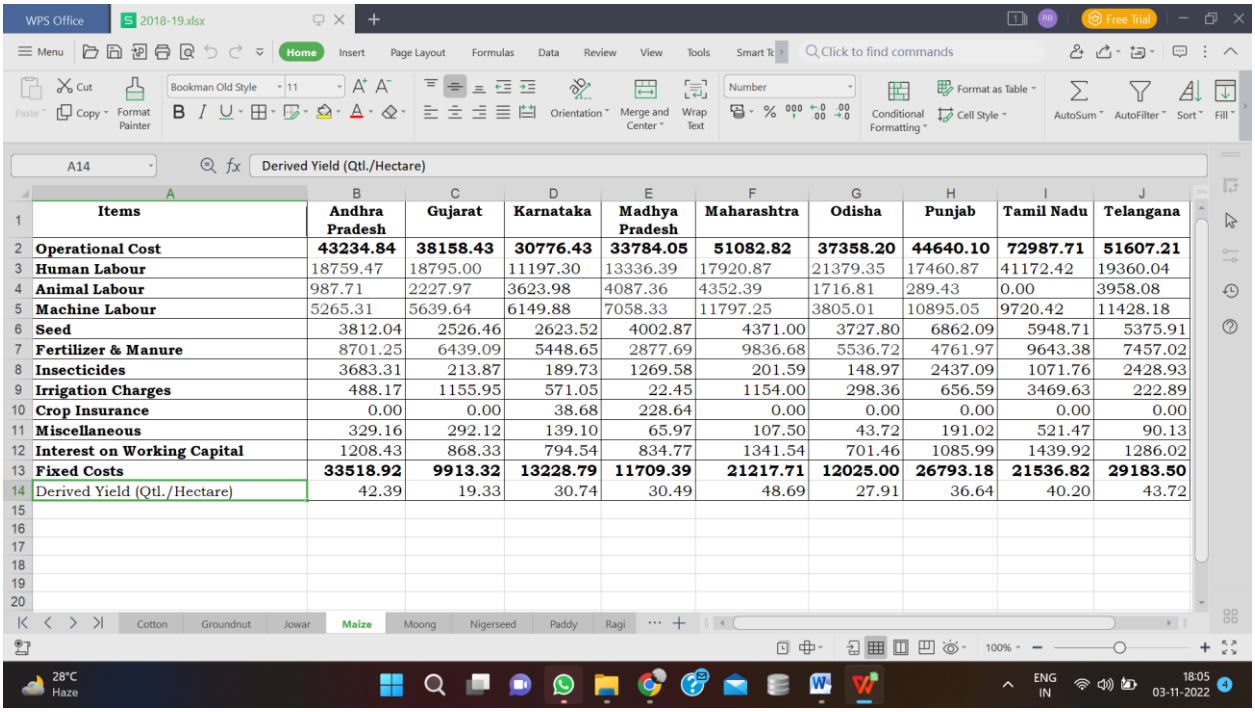
6.4. NEO4J

Neo4j is a graph database. A graph database, instead of having rows and columns has nodes edges and properties. It is more suitable for certain big data analytics applications than row and column databases or free-form JSON document databases for many use cases. A Graph database is used to represent relationships.

7. IMPLEMENTATION:

Here we took the data of three crops, Cotton, Paddy and Maize and predicted the Profit with the help of the data we had from IMD, crop models, cost of cultivation details and output price forecasts .

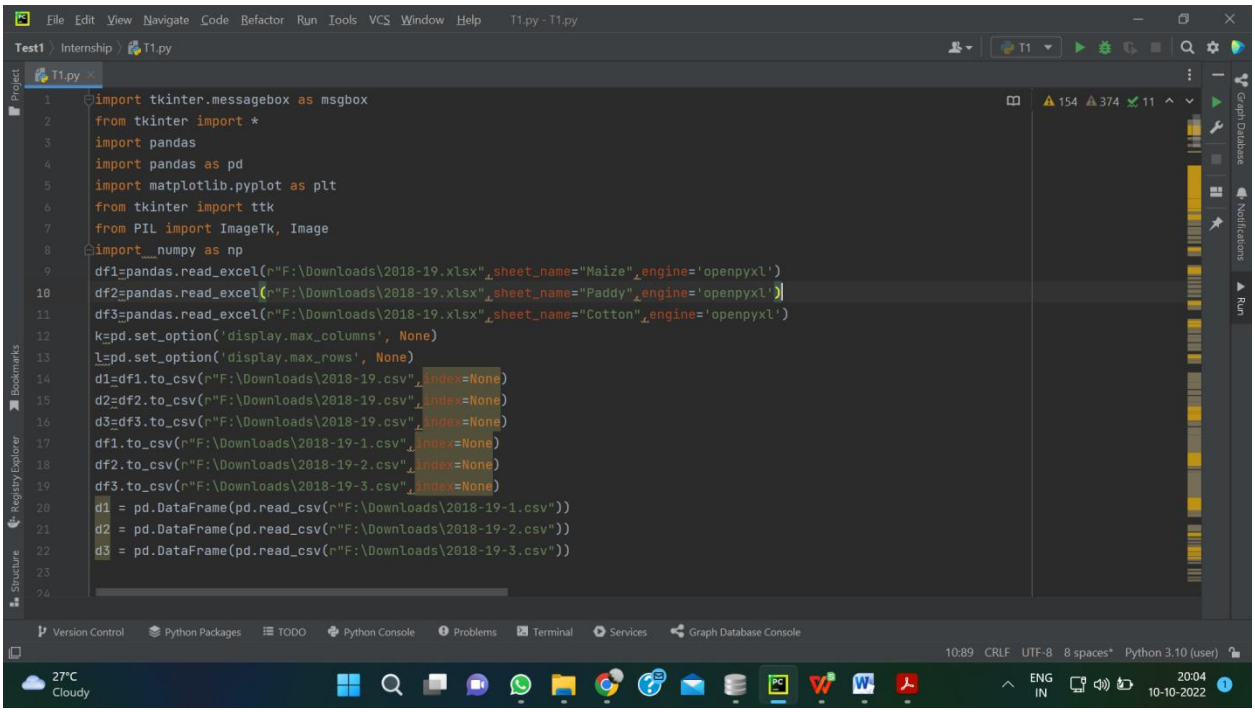
7.1. CODING



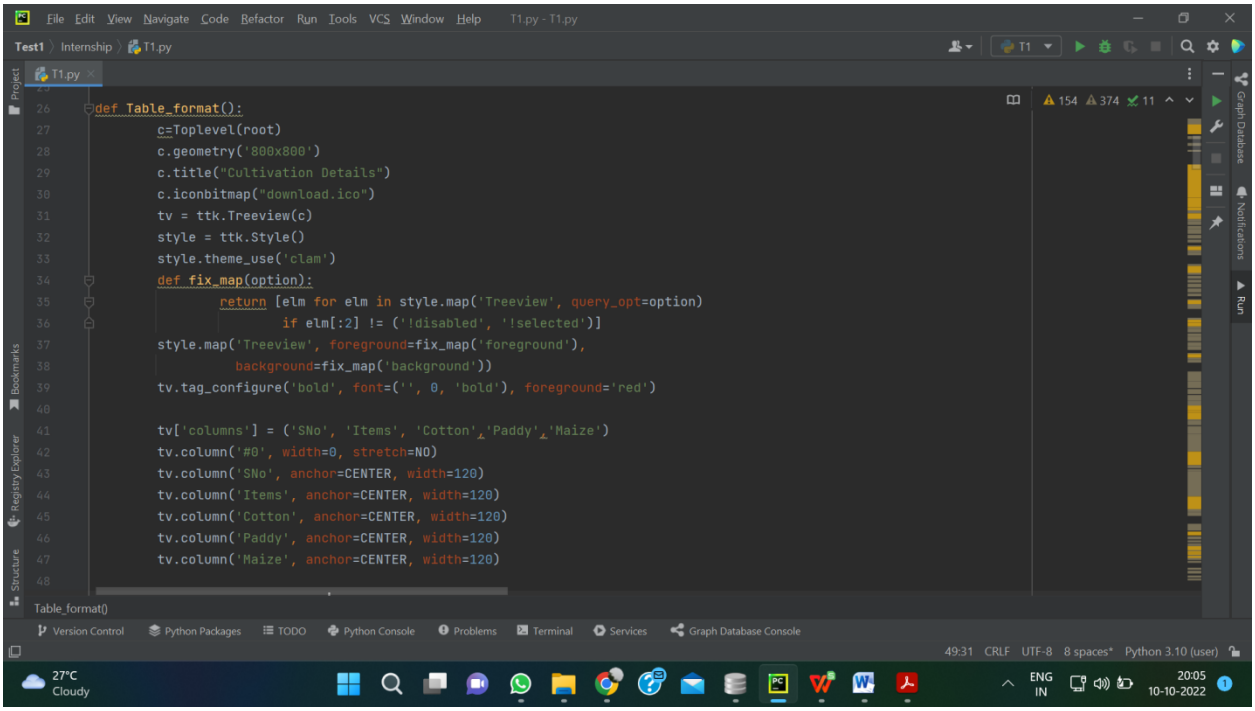
Items	Andhra Pradesh	Gujarat	Karnataka	Madhya Pradesh	Maharashtra	Odisha	Punjab	Tamil Nadu	Telangana
Operational Cost	**43234.84**	**38158.43**	**30776.43**	**33784.05**	**51082.82**	**37358.20**	**44640.10**	**72987.71**	**51607.21**
Human Labour	18759.47	18795.00	11197.30	13336.39	17920.87	21379.35	17460.87	41172.42	19360.04
Animal Labour	987.71	2227.97	3623.98	4087.36	4352.39	1716.81	289.43	0.00	3958.08
Machine Labour	5265.31	5639.64	6149.88	7058.33	11797.25	3805.01	10895.05	9720.42	11428.18
Seed	3812.04	2526.46	2623.52	4002.87	4371.00	3727.80	6862.09	5948.71	5375.91
Fertilizer & Manure	8701.25	6439.09	5448.65	2877.69	9836.68	5536.72	4761.97	9643.38	7457.02
Insecticides	3683.31	213.87	189.73	1269.58	201.59	148.97	2437.09	1071.76	2428.93
Irrigation Charges	488.17	1155.95	571.05	22.45	1154.00	298.36	656.59	3469.63	222.89
Crop Insurance	0.00	0.00	38.68	228.64	0.00	0.00	0.00	0.00	0.00
Miscellaneous	329.16	292.12	139.10	65.97	107.50	43.72	191.02	521.47	90.13
Interest on Working Capital	1208.43	868.33	794.54	834.77	1341.54	701.46	1085.99	1439.92	1286.02
Fixed Costs	**33518.92**	**9913.32**	**13228.79**	**11709.39**	**21217.71**	**12025.00**	**26793.18**	**21536.82**	**29183.50**
Derived Yield (Qtl./Hectare)	42.39	19.33	30.74	30.49	48.69	27.91	36.64	40.20	43.72

DATASET

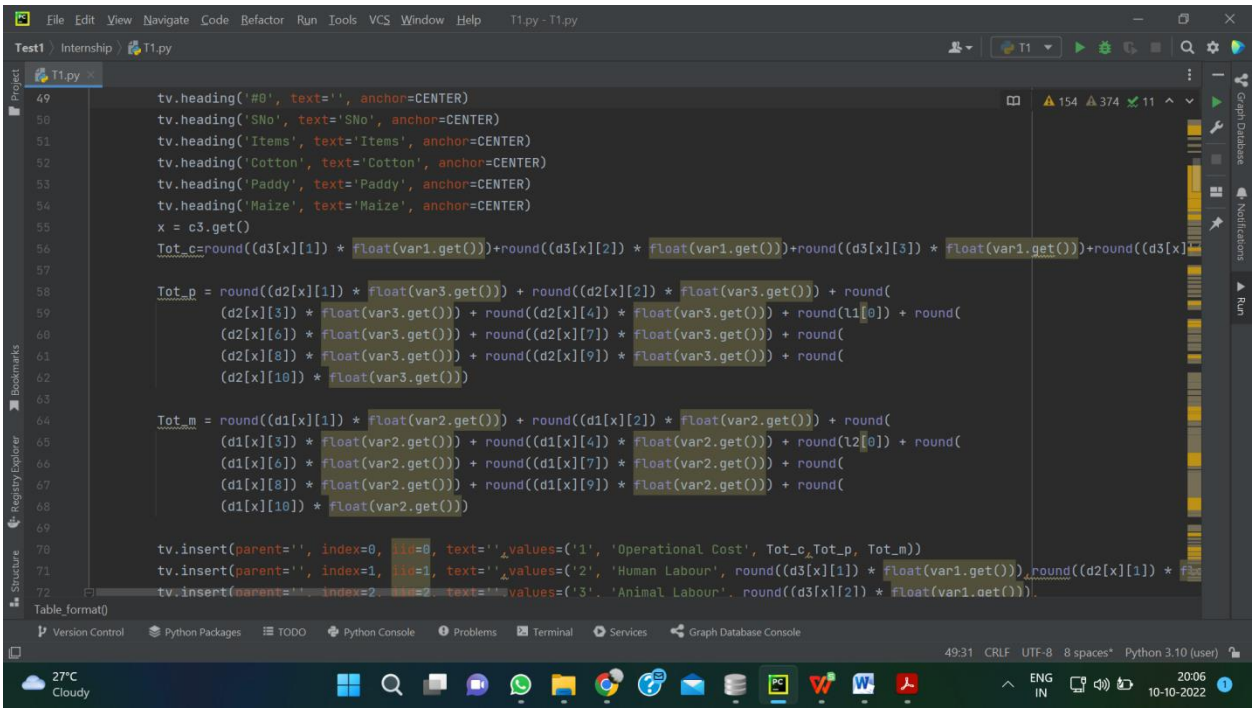
CODE:

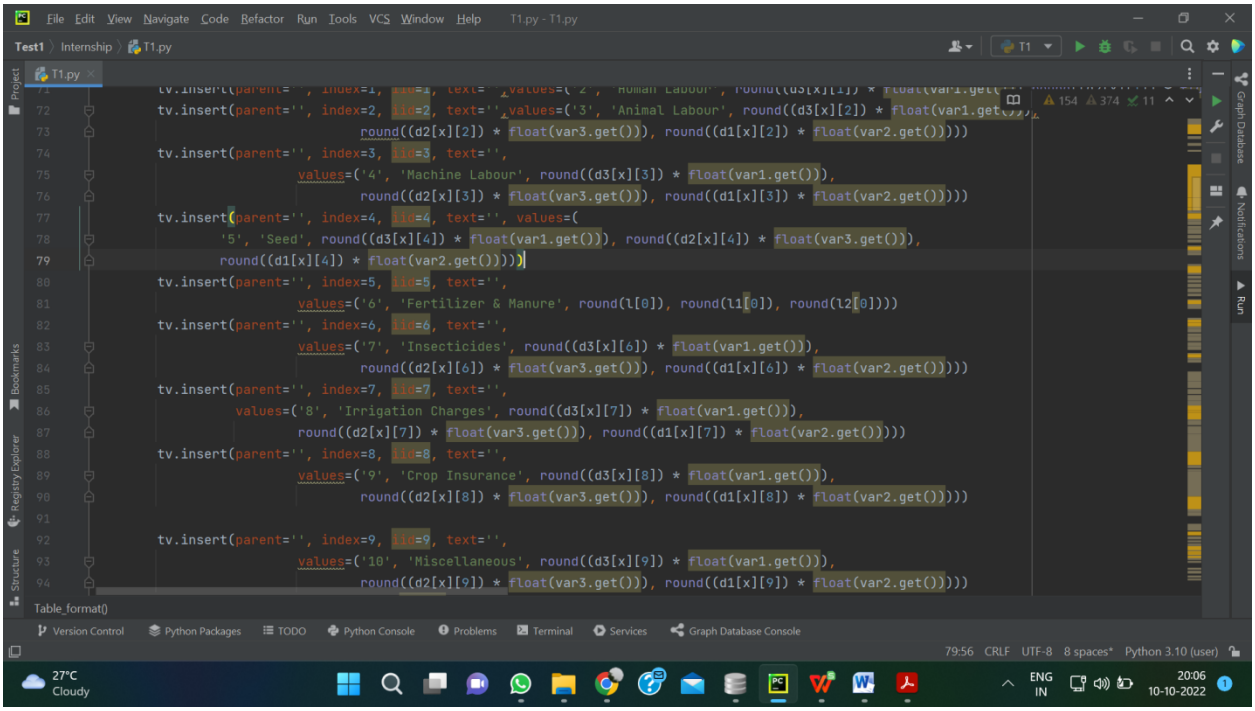


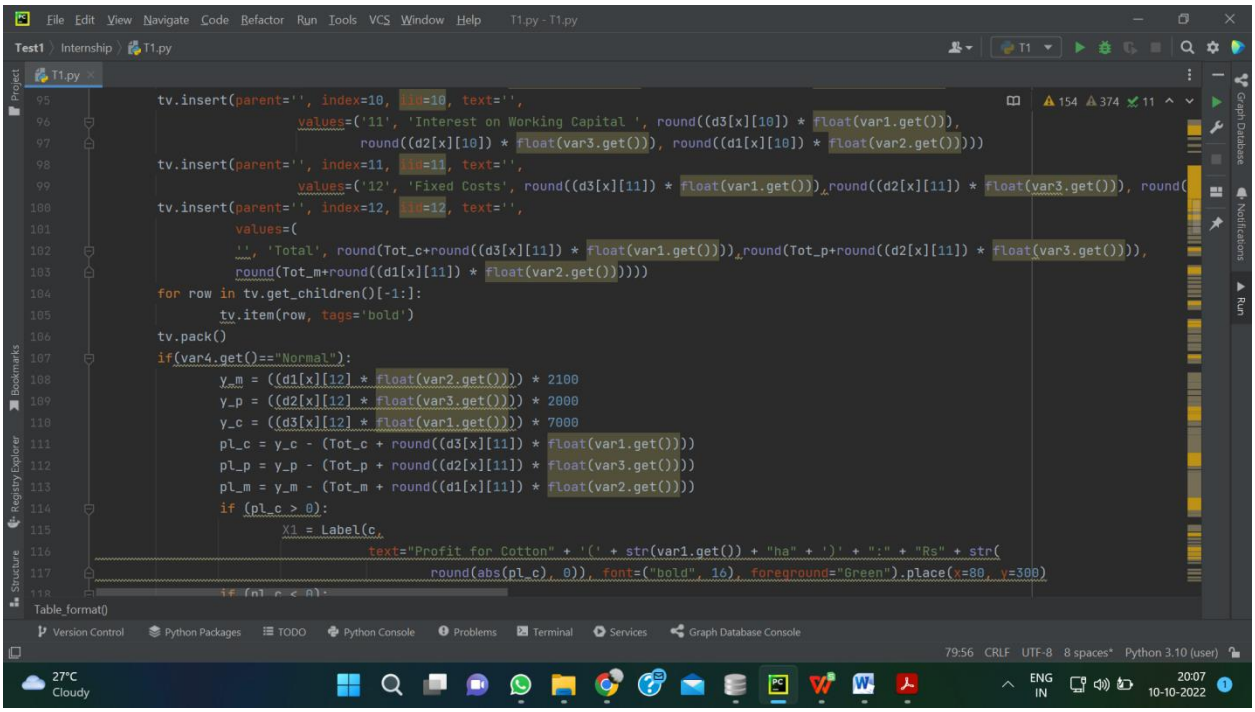
```
import tkinter.messagebox as msgbox
from tkinter import *
import pandas
import pandas as pd
import matplotlib.pyplot as plt
from tkinter import ttk
from PIL import ImageTk, Image
import numpy as np
df1=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Maize",engine='openpyxl')
df2=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Paddy",engine='openpyxl')
df3=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Cotton",engine='openpyxl')
k=pd.set_option('display.max_columns', None)
l=pd.set_option('display.max_rows', None)
d1=df1.to_csv(r"F:\Downloads\2018-19.csv",index=None)
d2=df2.to_csv(r"F:\Downloads\2018-19.csv",index=None)
d3=df3.to_csv(r"F:\Downloads\2018-19.csv",index=None)
df1.to_csv(r"F:\Downloads\2018-19-1.csv",index=None)
df2.to_csv(r"F:\Downloads\2018-19-2.csv",index=None)
df3.to_csv(r"F:\Downloads\2018-19-3.csv",index=None)
d1 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-1.csv"))
d2 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-2.csv"))
d3 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-3.csv"))
```



```
def Table_format():
        c=Toplevel(root)
        c.geometry('800x800')
        c.title("Cultivation Details")
        c.iconbitmap("download.ico")
        tv = ttk.Treeview(c)
        style = ttk.Style()
        style.theme_use('clam')
        def fix_map(option):
                return [elm for elm in style.map('Treeview', query_opt=option)
                        if elm[:2] != ('!disabled', '!selected')]
        style.map('Treeview', foreground=fix_map('foreground'),
                  background=fix_map('background'))
        tv.tag_configure('bold', font=('', 0, 'bold'), foreground='red')

        tv['columns'] = ('SNo', 'Items', 'Cotton','Paddy','Maize')
        tv.column('#0', width=0, stretch=NO)
        tv.column('SNo', anchor=CENTER, width=120)
        tv.column('Items', anchor=CENTER, width=120)
        tv.column('Cotton', anchor=CENTER, width=120)
        tv.column('Paddy', anchor=CENTER, width=120)
        tv.column('Maize', anchor=CENTER, width=120)
```

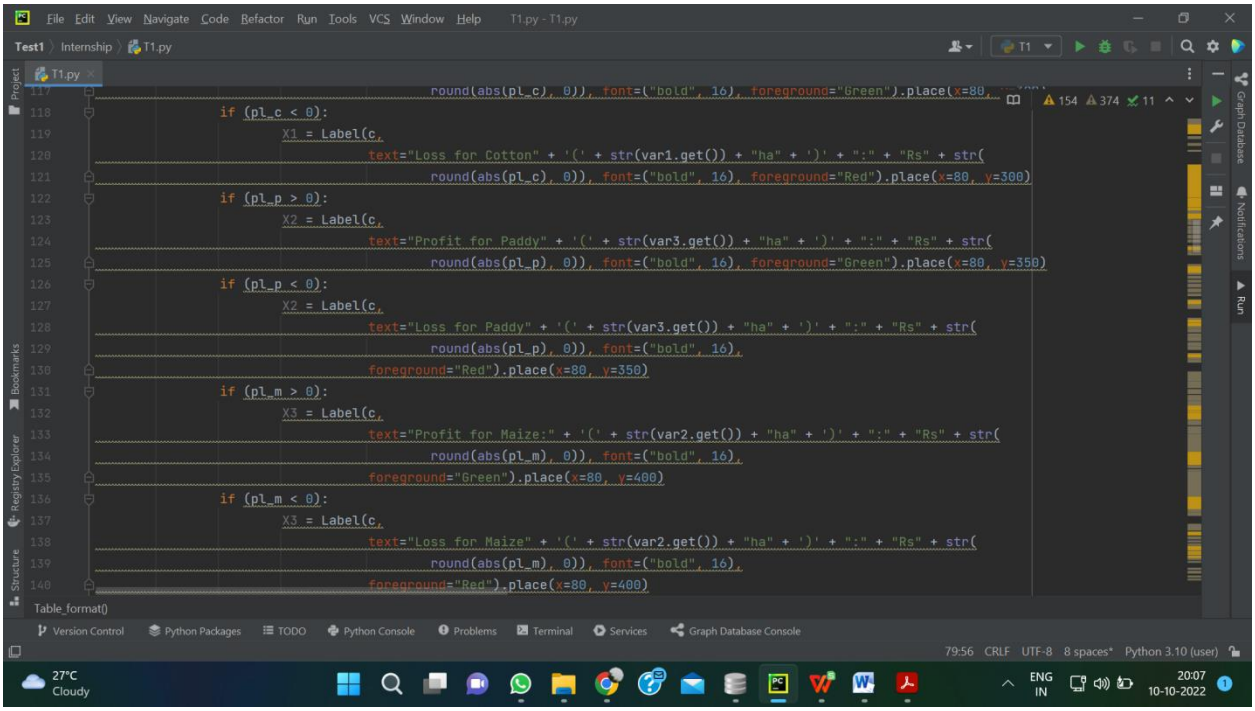

File Edit View Navigate Code Refactor Run Tools VCS Window Help
T1.py - T1.py
Test1 Internship T1.py
T1.py
tv.heading('#0', text='', anchor=CENTER)
tv.heading('SNo', text='SNo', anchor=CENTER)
tv.heading('Items', text='Items', anchor=CENTER)
tv.heading('Cotton', text='Cotton', anchor=CENTER)
tv.heading('Paddy', text='Paddy', anchor=CENTER)
tv.heading('Maize', text='Maize', anchor=CENTER)
x = c3.get()
Tot_c=round((d3[x][1]) * float(var1.get()))+round((d3[x][2]) * float(var1.get()))+round((d3[x][3]) * float(var1.get()))+round((d3[x]
Tot_p = round((d2[x][1]) * float(var3.get())) + round((d2[x][2]) * float(var3.get())) + round(
(d2[x][3]) * float(var3.get())) + round((d2[x][4]) * float(var3.get())) + round(l1[0]) + round(
(d2[x][6]) * float(var3.get())) + round((d2[x][7]) * float(var3.get())) + round(
(d2[x][8]) * float(var3.get())) + round((d2[x][9]) * float(var3.get())) + round(
(d2[x][10]) * float(var3.get()))
Tot_m = round((d1[x][1]) * float(var2.get())) + round((d1[x][2]) * float(var2.get())) + round(
(d1[x][3]) * float(var2.get())) + round((d1[x][4]) * float(var2.get())) + round(l2[0]) + round(
(d1[x][6]) * float(var2.get())) + round((d1[x][7]) * float(var2.get())) + round(
(d1[x][8]) * float(var2.get())) + round((d1[x][9]) * float(var2.get())) + round(
(d1[x][10]) * float(var2.get()))
tv.insert(parent='', index=0, iid=0, text='',values=('1', 'Operational Cost', Tot_c,Tot_p, Tot_m))
tv.insert(parent='', index=1, iid=1, text='',values=('2', 'Human Labour', round((d3[x][1]) * float(var1.get())),round((d2[x][1]) * f
tv.insert(parent='', index=2, iid=2, text='',values=('3', 'Animal Labour', round((d3[x][2]) * float(var1.get()))
Table_format()
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Graph Database Console
49:31 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:06 10-10-2022


File Edit View Navigate Code Refactor Run Tools VCS Window Help
T1.py - T1.py
Test1 Internship T1.py
T1.py
tv.insert(parent='', index=2, iid=2, text='',values=('3', 'Animal Labour', round((d3[x][2]) * float(var1.get()))),
round((d2[x][2]) * float(var3.get())), round((d1[x][2]) * float(var2.get()))))
tv.insert(parent='', index=3, iid=3, text='',
values=('4', 'Machine Labour', round((d3[x][3]) * float(var1.get())),
round((d2[x][3]) * float(var3.get())), round((d1[x][3]) * float(var2.get()))))
tv.insert(parent='', index=4, iid=4, text='', values=(
'5', 'Seed', round((d3[x][4]) * float(var1.get())), round((d2[x][4]) * float(var3.get())),
round((d1[x][4]) * float(var2.get()))))
tv.insert(parent='', index=5, iid=5, text='',
values=('6', 'Fertilizer & Manure', round(l1[0]), round(l1[0]), round(l2[0])))
tv.insert(parent='', index=6, iid=6, text='',
values=('7', 'Insecticides', round((d3[x][6]) * float(var1.get())),
round((d2[x][6]) * float(var3.get())), round((d1[x][6]) * float(var2.get()))))
tv.insert(parent='', index=7, iid=7, text='',
values=('8', 'Irrigation Charges', round((d3[x][7]) * float(var1.get())),
round((d2[x][7]) * float(var3.get())), round((d1[x][7]) * float(var2.get()))))
tv.insert(parent='', index=8, iid=8, text='',
values=('9', 'Crop Insurance', round((d3[x][8]) * float(var1.get())),
round((d2[x][8]) * float(var3.get())), round((d1[x][8]) * float(var2.get()))))
tv.insert(parent='', index=9, iid=9, text='',
values=('10', 'Miscellaneous', round((d3[x][9]) * float(var1.get())),
round((d2[x][9]) * float(var3.get())), round((d1[x][9]) * float(var2.get()))))
Table_format()
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Graph Database Console
79:56 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:06 10-10-2022



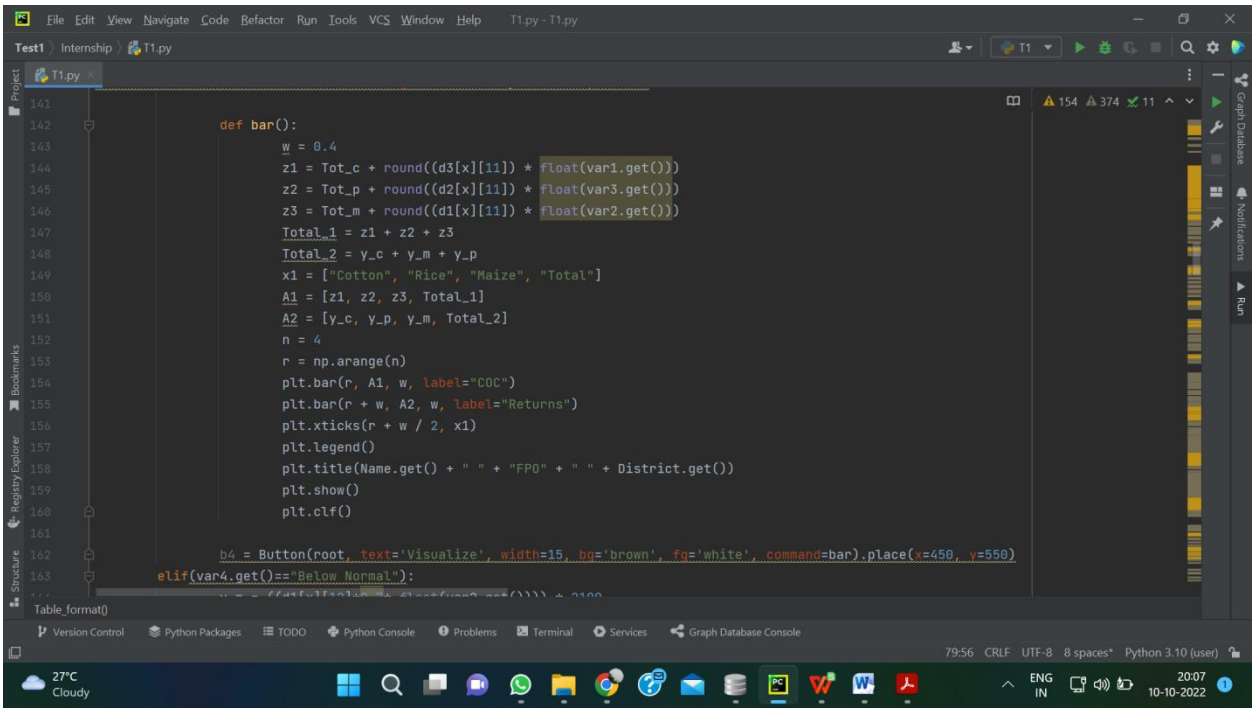
```
        tv.insert(parent='', index=10, iid=10, text='',
                  values=('11', 'Interest on Working Capital ', round((d3[x][10]) * float(var1.get())),
                          round((d2[x][10]) * float(var3.get())), round((d1[x][10]) * float(var2.get()))))
        tv.insert(parent='', index=11, iid=11, text='',
                  values=('12', 'Fixed Costs', round((d3[x][11]) * float(var1.get())),round((d2[x][11]) * float(var3.get())), round(
        tv.insert(parent='', index=12, iid=12, text='',
                  values=(
                  '', 'Total', round(Tot_c+round((d3[x][11]) * float(var1.get()))),round(Tot_p+round((d2[x][11]) * float(var3.get()))),
                  round(Tot_m+round((d1[x][11]) * float(var2.get())))))
        for row in tv.get_children()[-1:]:
                tv.item(row, tags='bold')
        tv.pack()
        if(var4.get()=="Normal"):
                y_m = ((d1[x][12] * float(var2.get()))) * 2100
                y_p = ((d2[x][12] * float(var3.get()))) * 2000
                y_c = ((d3[x][12] * float(var1.get()))) * 7000
                pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
                pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
                pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
                if (pl_c > 0):
                        X1 = Label(c,
                                   text="Profit for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_c), 0)), font=("bold", 16), foreground="Green").place(x=80, y=300)
                if (pl_c < 0):
```

Table_format()

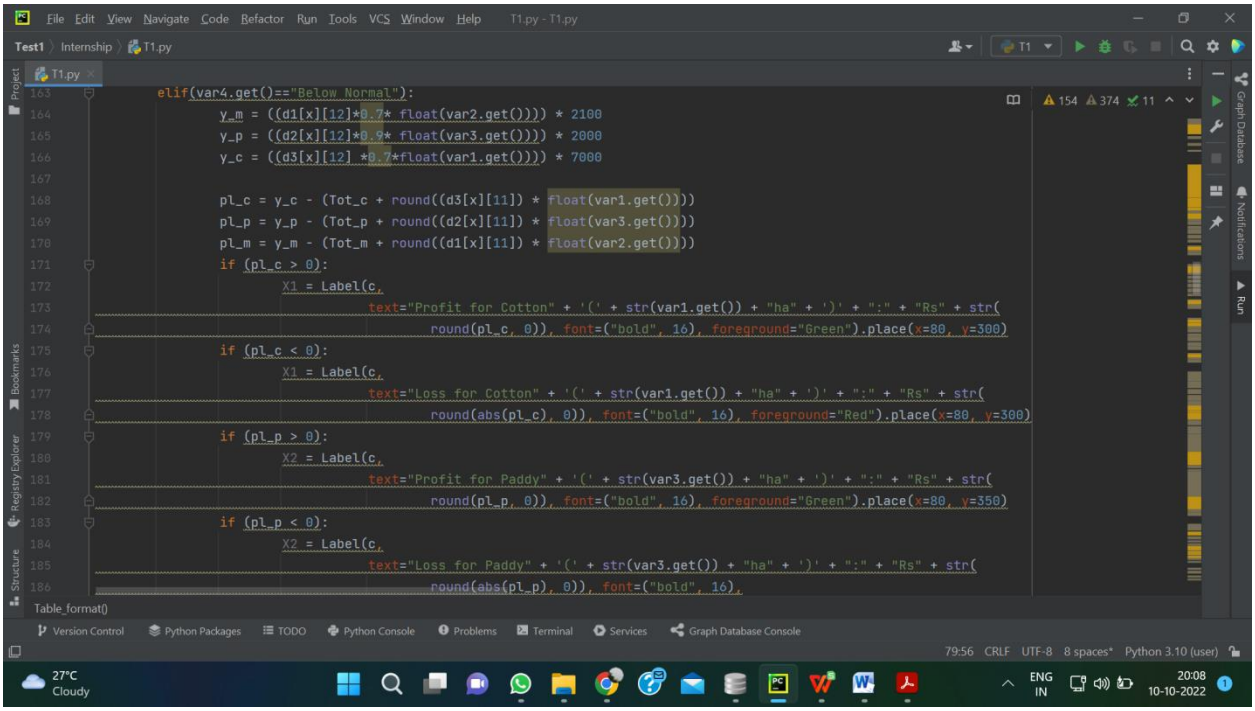


```
                                           round(abs(pl_c), 0)), font=("bold", 16), foreground="Green").place(x=80,
                if (pl_c < 0):
                        X1 = Label(c,
                                   text="Loss for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_c), 0)), font=("bold", 16), foreground="Red").place(x=80, y=300)
                if (pl_p > 0):
                        X2 = Label(c,
                                   text="Profit for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_p), 0)), font=("bold", 16), foreground="Green").place(x=80, y=350)
                if (pl_p < 0):
                        X2 = Label(c,
                                   text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_p), 0)), font=("bold", 16),
                                   foreground="Red").place(x=80, y=350)
                if (pl_m > 0):
                        X3 = Label(c,
                                   text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_m), 0)), font=("bold", 16),
                                   foreground="Green").place(x=80, y=400)
                if (pl_m < 0):
                        X3 = Label(c,
                                   text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                                           round(abs(pl_m), 0)), font=("bold", 16),
                                   foreground="Red").place(x=80, y=400)
```

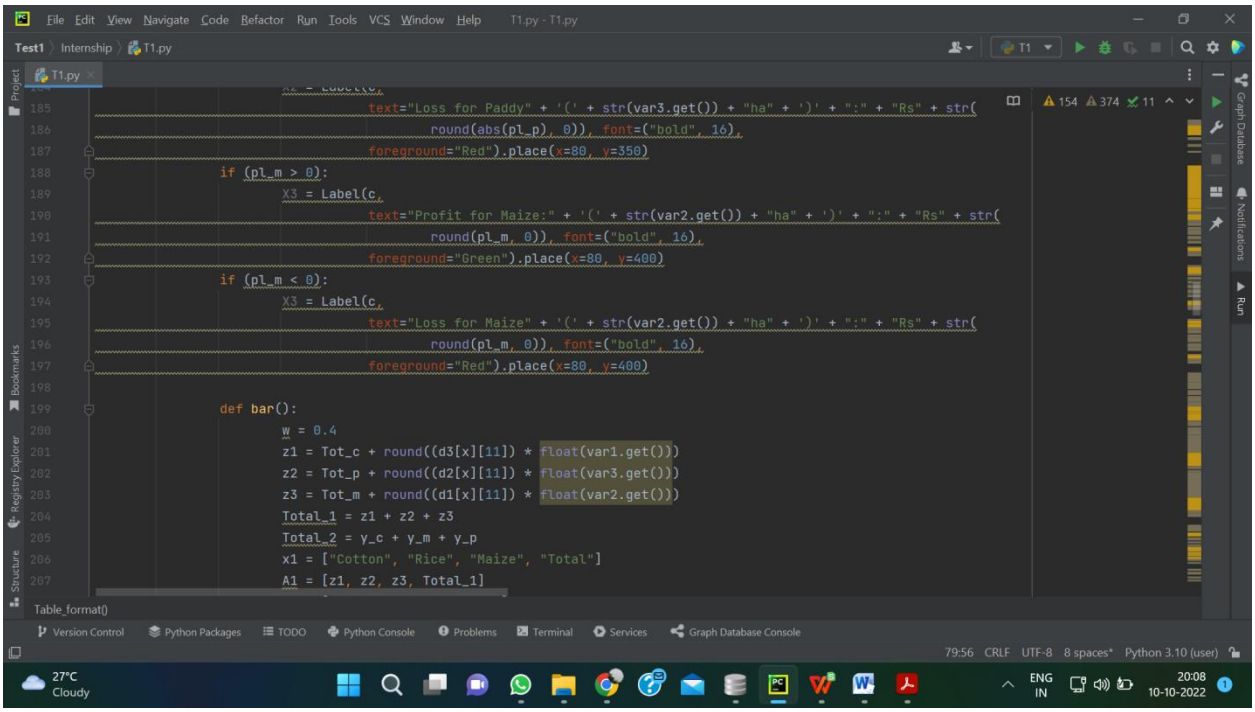
Table_format()

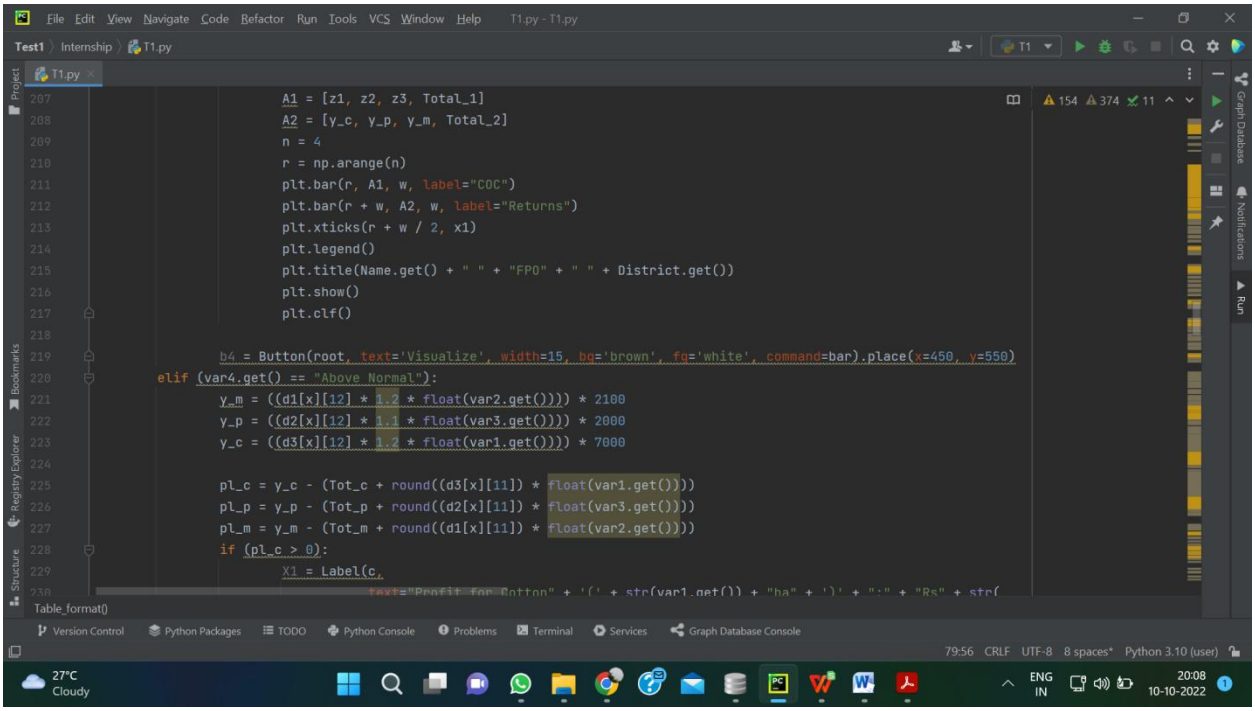


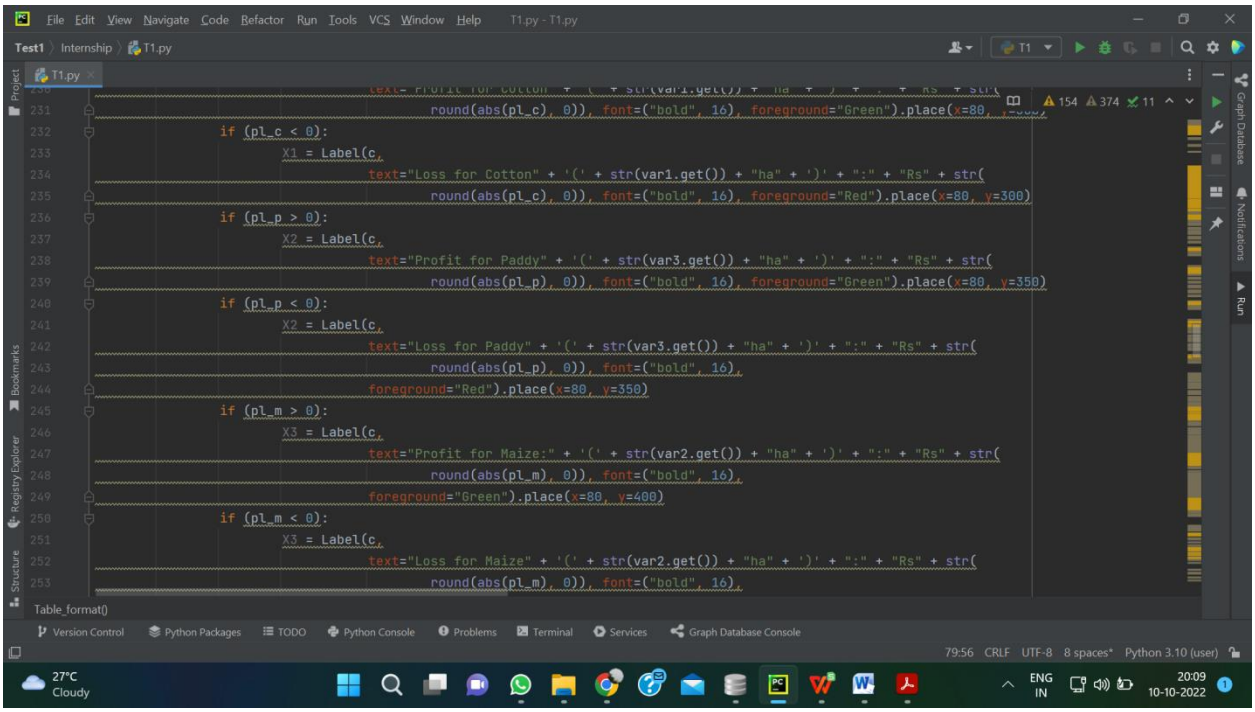
```
def bar():
        w = 0.4
        z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
        z2 = Tot_p + round((d2[x][11]) * float(var3.get()))
        z3 = Tot_m + round((d1[x][11]) * float(var2.get()))
        Total_1 = z1 + z2 + z3
        Total_2 = y_c + y_m + y_p
        x1 = ["Cotton", "Rice", "Maize", "Total"]
        A1 = [z1, z2, z3, Total_1]
        A2 = [y_c, y_p, y_m, Total_2]
        n = 4
        r = np.arange(n)
        plt.bar(r, A1, w, label="COC")
        plt.bar(r + w, A2, w, label="Returns")
        plt.xticks(r + w / 2, x1)
        plt.legend()
        plt.title(Name.get() + " " + "FPO" + " " + District.get())
        plt.show()
        plt.clf()

    b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
elif(var4.get()=="Below Normal"):
```

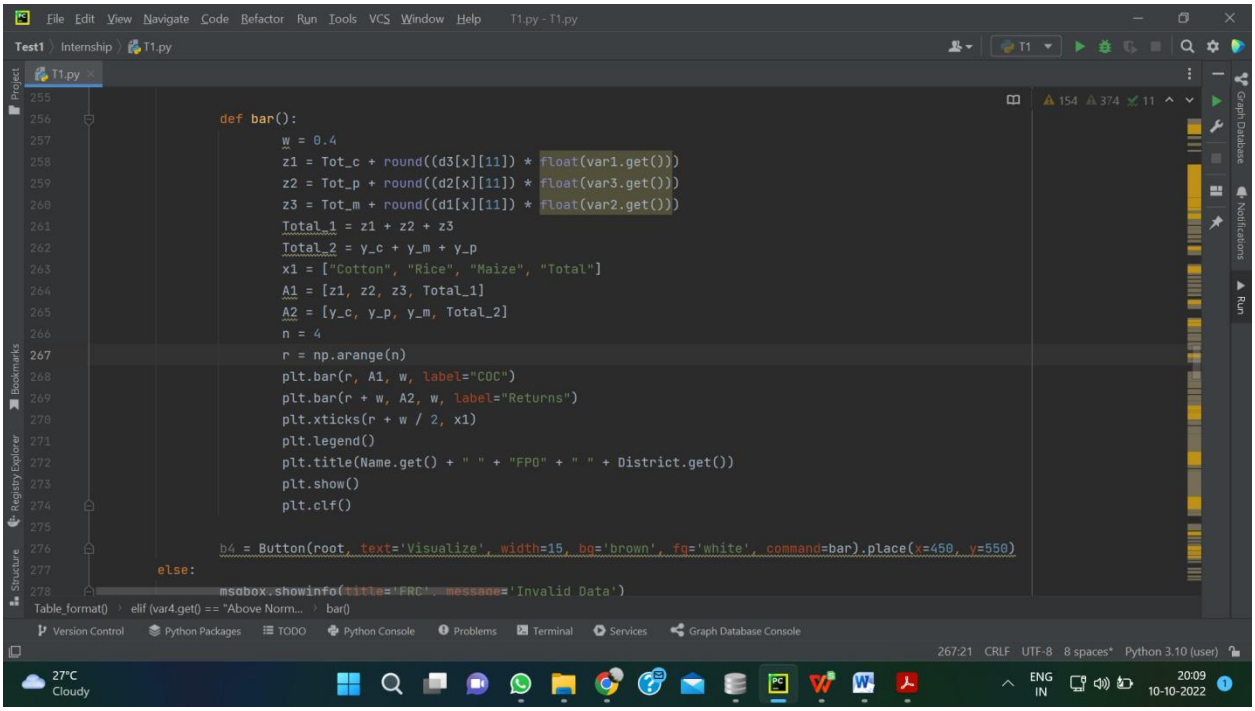


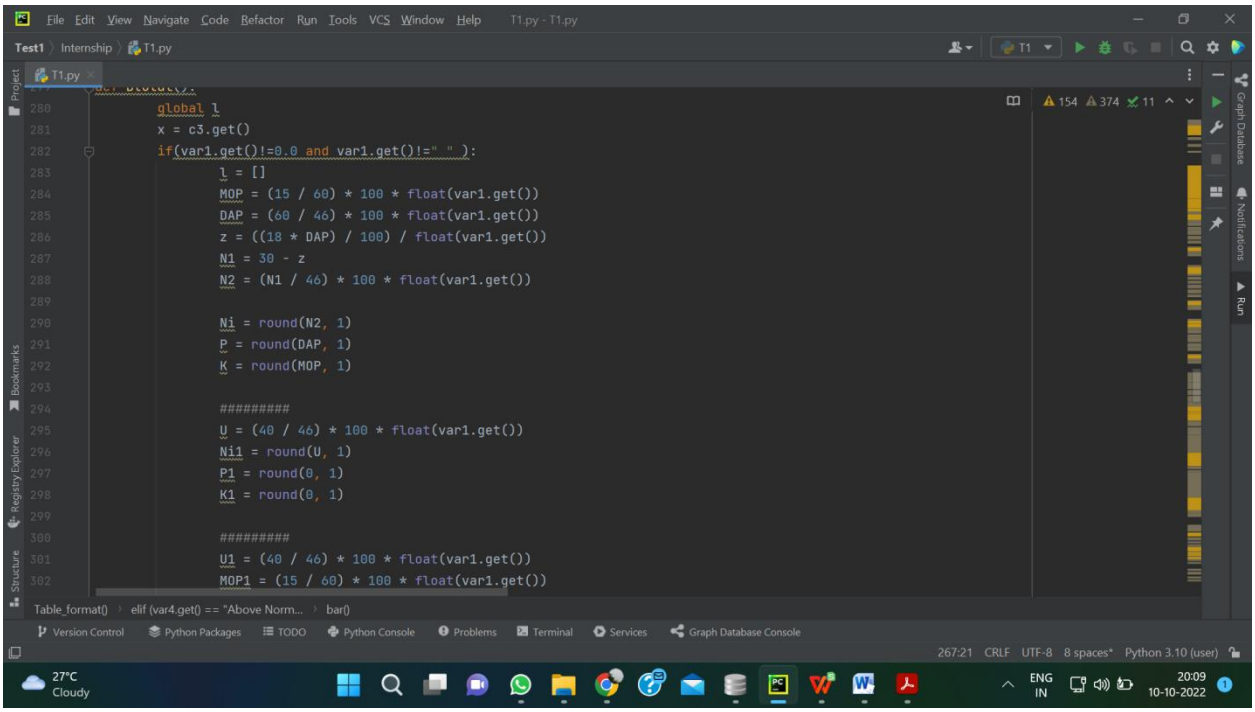
```
elif(var4.get()=="Below Normal"):
        y_m = ((d1[x][12]*0.7* float(var2.get()))) * 2100
        y_p = ((d2[x][12]*0.9* float(var3.get()))) * 2000
        y_c = ((d3[x][12] *0.7*float(var1.get()))) * 7000

        pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
        pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
        pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
        if (pl_c > 0):
            X1 = Label(c,
                       text="Profit for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                           round(pl_c, 0)), font=("bold", 16), foreground="Green").place(x=80, y=300)
        if (pl_c < 0):
            X1 = Label(c,
                       text="Loss for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                           round(abs(pl_c), 0)), font=("bold", 16), foreground="Red").place(x=80, y=300)
        if (pl_p > 0):
            X2 = Label(c,
                       text="Profit for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                           round(pl_p, 0)), font=("bold", 16), foreground="Green").place(x=80, y=350)
        if (pl_p < 0):
            X2 = Label(c,
                       text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                           round(abs(pl_p), 0)), font=("bold", 16),
```

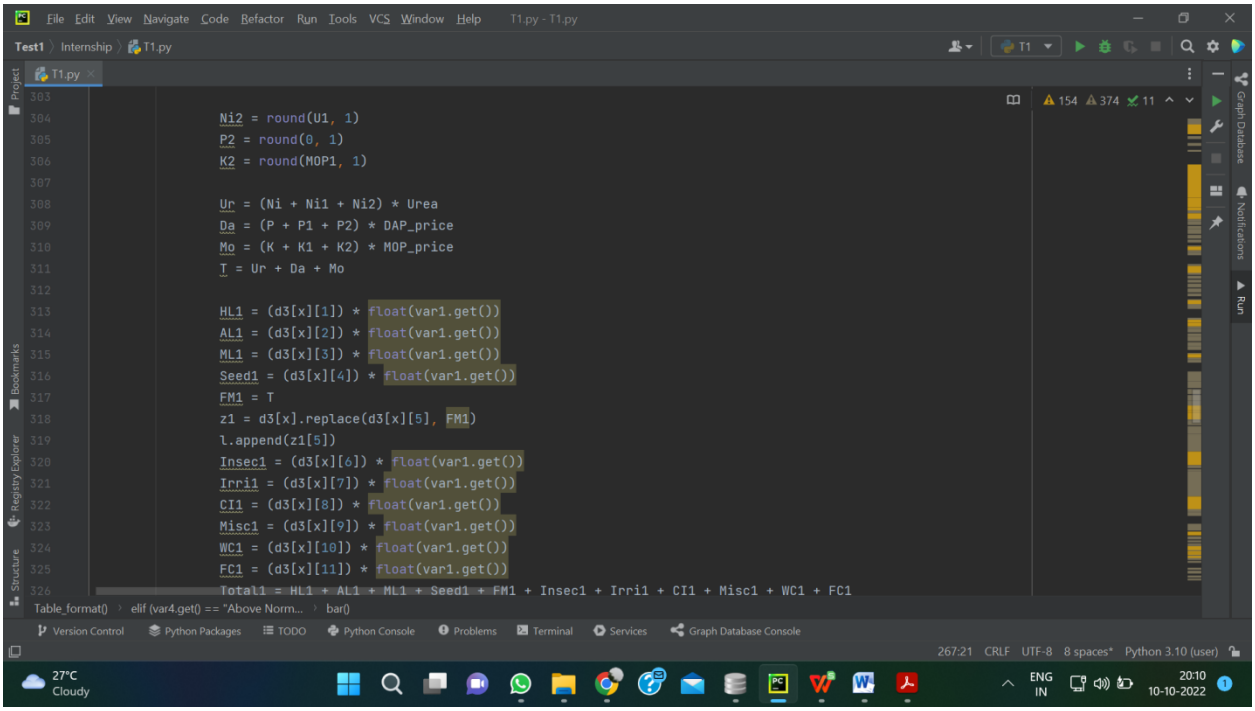

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_p), 0)), font=("bold", 16),
foreground="Red").place(x=80, y=350)
if (pl_m > 0):
X3 = Label(c,
text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
round(pl_m, 0)), font=("bold", 16),
foreground="Green").place(x=80, y=400)
if (pl_m < 0):
X3 = Label(c,
text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
round(pl_m, 0)), font=("bold", 16),
foreground="Red").place(x=80, y=400)
def bar():
w = 0.4
z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
z2 = Tot_p + round((d2[x][11]) * float(var3.get()))
z3 = Tot_m + round((d1[x][11]) * float(var2.get()))
Total_1 = z1 + z2 + z3
Total_2 = y_c + y_m + y_p
x1 = ["Cotton", "Rice", "Maize", "Total"]
A1 = [z1, z2, z3, Total_1]
Table_format()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
79:56 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:08 10-10-2022

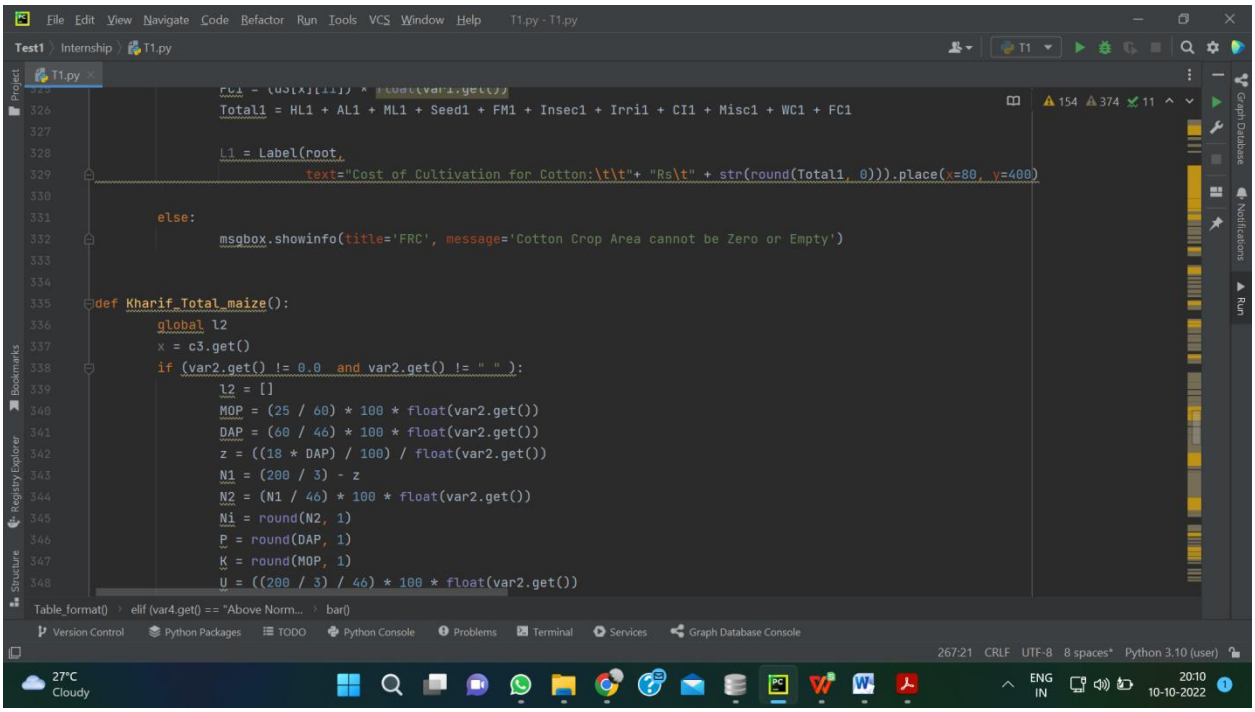

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
A1 = [z1, z2, z3, Total_1]
A2 = [y_c, y_p, y_m, Total_2]
n = 4
r = np.arange(n)
plt.bar(r, A1, w, label="COC")
plt.bar(r + w, A2, w, label="Returns")
plt.xticks(r + w / 2, x1)
plt.legend()
plt.title(Name.get() + " " + "FPO" + " " + District.get())
plt.show()
plt.clf()
b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
elif (var4.get() == "Above Normal"):
y_m = ((d1[x][12] * 1.2 * float(var2.get()))) * 2100
y_p = ((d2[x][12] * 1.1 * float(var3.get()))) * 2000
y_c = ((d3[x][12] * 1.2 * float(var1.get()))) * 7000
pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
if (pl_c > 0):
X1 = Label(c,
Table_format()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
79:56 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:08 10-10-2022

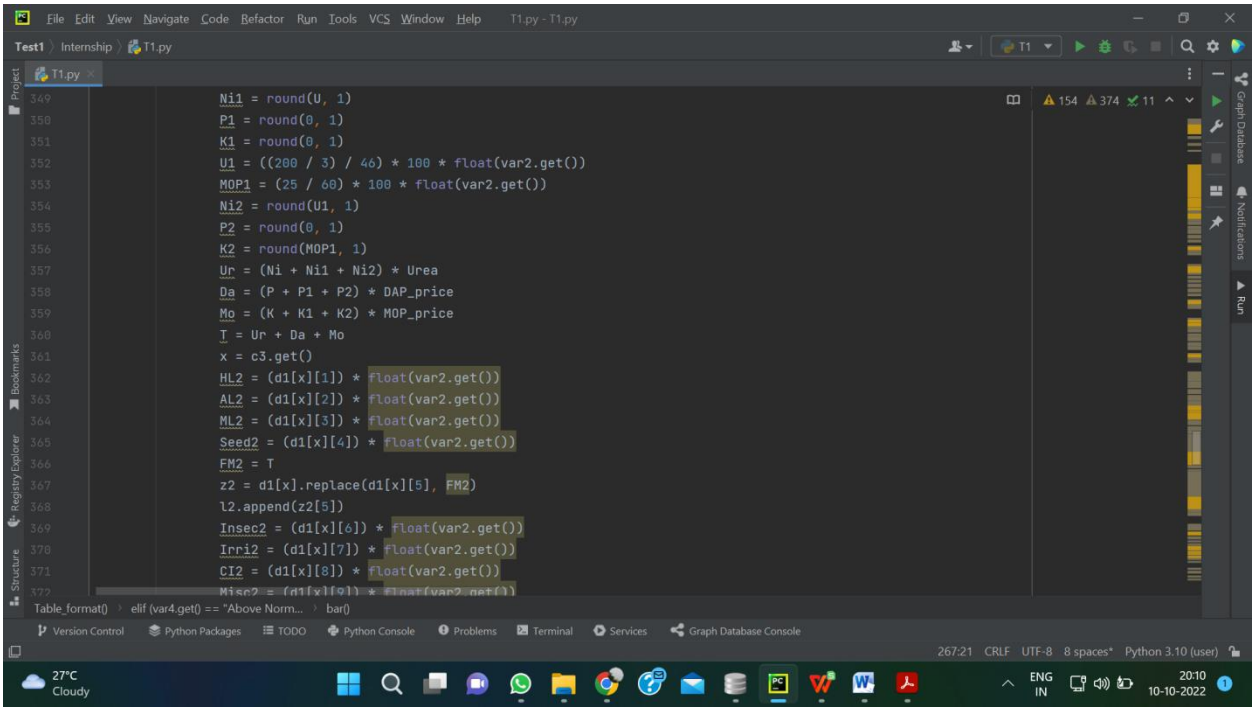

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
round(abs(pl_c), 0)), font=("bold", 16), foreground="Green").place(x=80, y=300)
if (pl_c < 0):
X1 = Label(c,
text="Loss for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_c), 0)), font=("bold", 16), foreground="Red").place(x=80, y=300)
if (pl_p > 0):
X2 = Label(c,
text="Profit for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_p), 0)), font=("bold", 16), foreground="Green").place(x=80, y=350)
if (pl_p < 0):
X2 = Label(c,
text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_p), 0)), font=("bold", 16),
foreground="Red").place(x=80, y=350)
if (pl_m > 0):
X3 = Label(c,
text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_m), 0)), font=("bold", 16),
foreground="Green").place(x=80, y=400)
if (pl_m < 0):
X3 = Label(c,
text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
round(abs(pl_m), 0)), font=("bold", 16),
Table_format()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
79:56 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:09 10-10-2022

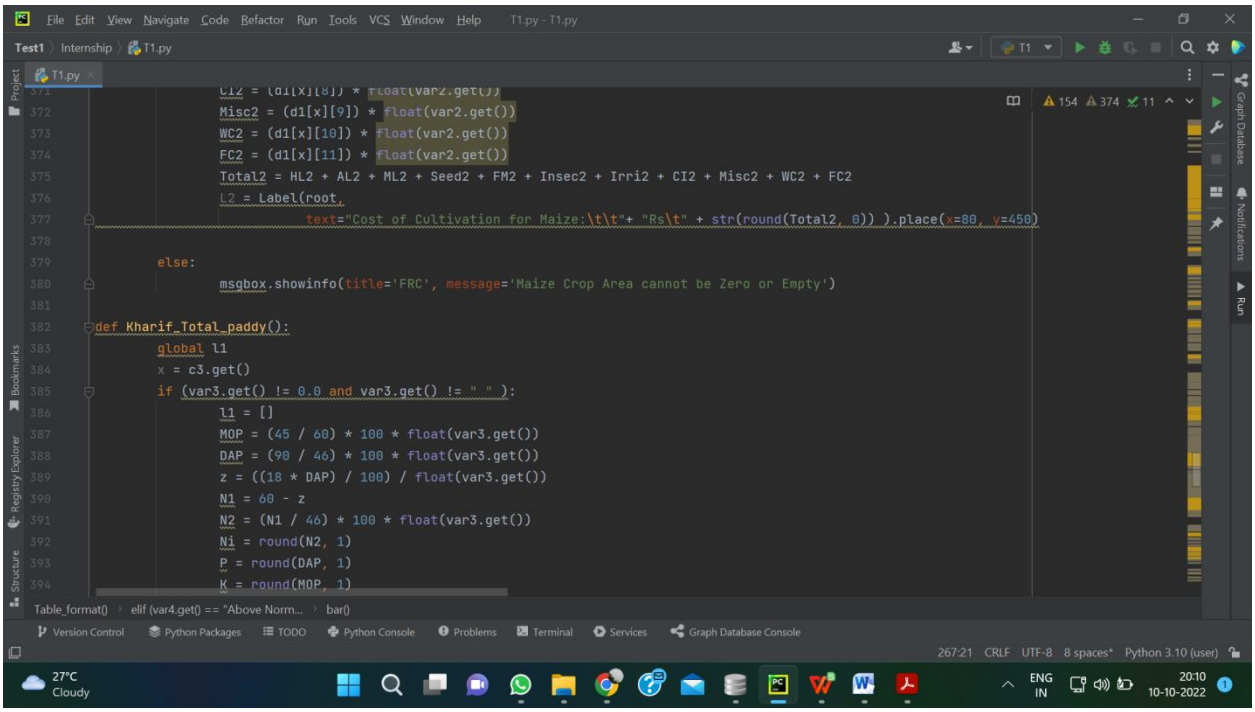

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
def bar():
w = 0.4
z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
z2 = Tot_p + round((d2[x][11]) * float(var3.get()))
z3 = Tot_m + round((d1[x][11]) * float(var2.get()))
Total_1 = z1 + z2 + z3
Total_2 = y_c + y_m + y_p
x1 = ["Cotton", "Rice", "Maize", "Total"]
A1 = [z1, z2, z3, Total_1]
A2 = [y_c, y_p, y_m, Total_2]
n = 4
r = np.arange(n)
plt.bar(r, A1, w, label="COC")
plt.bar(r + w, A2, w, label="Returns")
plt.xticks(r + w / 2, x1)
plt.legend()
plt.title(Name.get() + " " + "FPO" + " " + District.get())
plt.show()
plt.clf()
b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
else:
msgbox.showinfo(title='FRC', message='Invalid Data')
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:09 10-10-2022

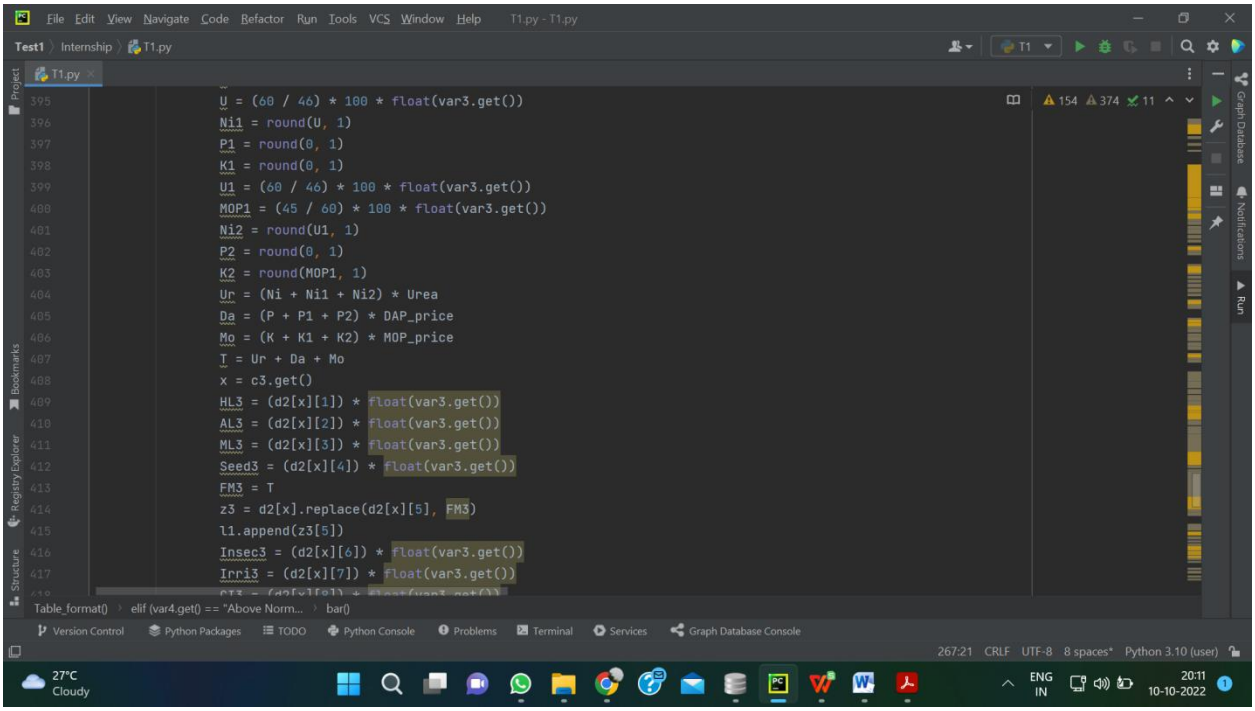

File Edit View Navigate Code Refactor Run Tools VCS Window Help
T1.py - T1.py
Test1 Internship T1.py
T1.py
global l
x = c3.get()
if(var1.get()!=0.0 and var1.get()!=" "):
l = []
MOP = (15 / 60) * 100 * float(var1.get())
DAP = (60 / 46) * 100 * float(var1.get())
z = ((18 * DAP) / 100) / float(var1.get())
N1 = 30 - z
N2 = (N1 / 46) * 100 * float(var1.get())
Ni = round(N2, 1)
P = round(DAP, 1)
K = round(MOP, 1)
#########
U = (40 / 46) * 100 * float(var1.get())
Ni1 = round(U, 1)
P1 = round(0, 1)
K1 = round(0, 1)
#########
U1 = (40 / 46) * 100 * float(var1.get())
MOP1 = (15 / 60) * 100 * float(var1.get())
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:09 10-10-2022

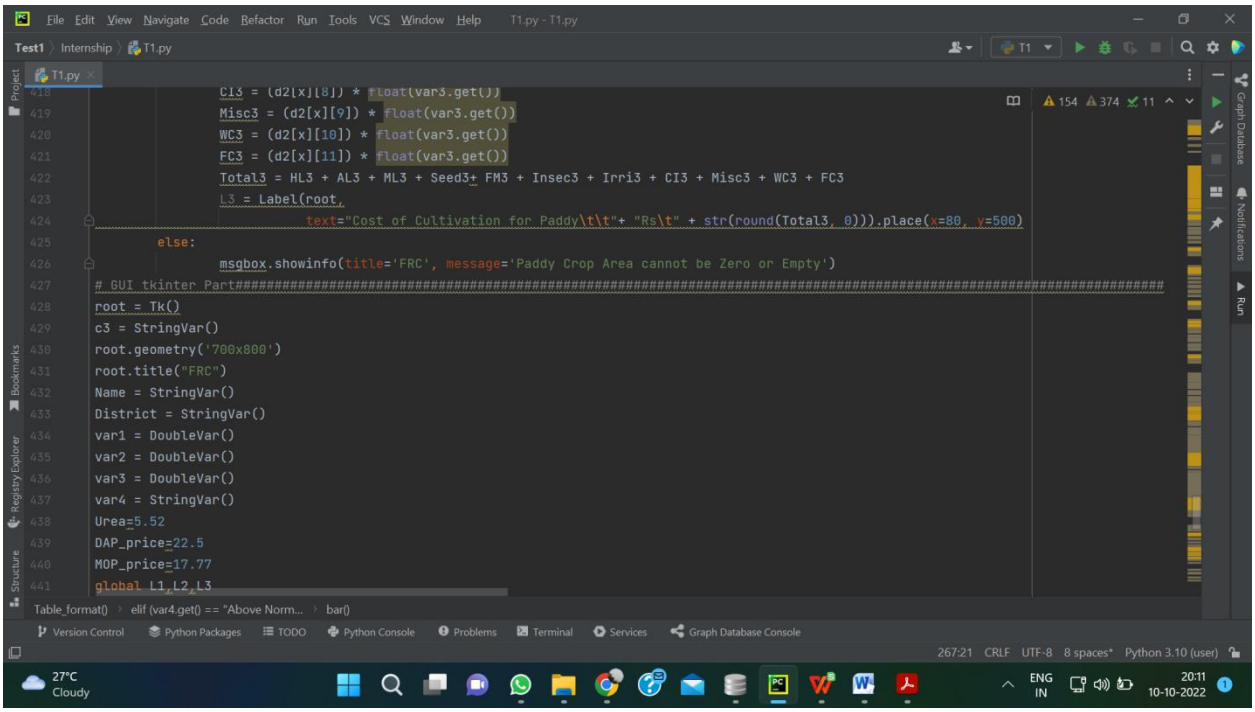

File Edit View Navigate Code Refactor Run Tools VCS Window Help
T1.py - T1.py
Test1 Internship T1.py
T1.py
Ni2 = round(U1, 1)
P2 = round(0, 1)
K2 = round(MOP1, 1)
Ur = (Ni + Ni1 + Ni2) * Urea
Da = (P + P1 + P2) * DAP_price
Mo = (K + K1 + K2) * MOP_price
T = Ur + Da + Mo
HL1 = (d3[x][1]) * float(var1.get())
AL1 = (d3[x][2]) * float(var1.get())
ML1 = (d3[x][3]) * float(var1.get())
Seed1 = (d3[x][4]) * float(var1.get())
FM1 = T
z1 = d3[x].replace(d3[x][5], FM1)
l.append(z1[5])
Insec1 = (d3[x][6]) * float(var1.get())
Irri1 = (d3[x][7]) * float(var1.get())
CI1 = (d3[x][8]) * float(var1.get())
Misc1 = (d3[x][9]) * float(var1.get())
WC1 = (d3[x][10]) * float(var1.get())
FC1 = (d3[x][11]) * float(var1.get())
Total1 = HL1 + AL1 + ML1 + Seed1 + FM1 + Insec1 + Irri1 + CI1 + Misc1 + WC1 + FC1
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:10 10-10-2022


File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
Total1 = HL1 + AL1 + ML1 + Seed1 + FM1 + Insec1 + Irri1 + CI1 + Misc1 + WC1 + FC1
L1 = Label(root,
text="Cost of Cultivation for Cotton:\t\t"+ "Rs\t" + str(round(Total1, 0))).place(x=80, y=400)
else:
msgbox.showinfo(title='FRC', message='Cotton Crop Area cannot be Zero or Empty')
def Kharif_Total_maize():
global l2
x = c3.get()
if (var2.get() != 0.0 and var2.get() != " "):
l2 = []
MOP = (25 / 60) * 100 * float(var2.get())
DAP = (60 / 46) * 100 * float(var2.get())
z = ((18 * DAP) / 100) / float(var2.get())
N1 = (200 / 3) - z
N2 = (N1 / 46) * 100 * float(var2.get())
Ni = round(N2, 1)
P = round(DAP, 1)
K = round(MOP, 1)
U = ((200 / 3) / 46) * 100 * float(var2.get())
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:10 10-10-2022

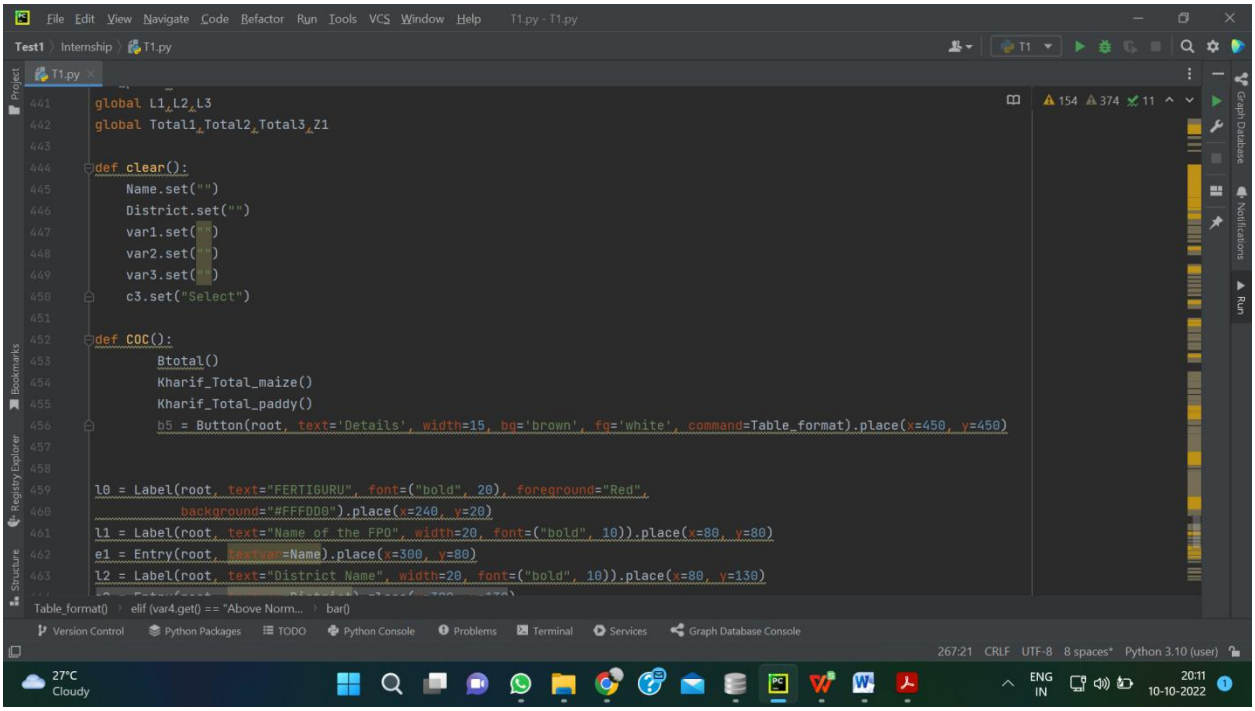

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
Ni1 = round(U, 1)
P1 = round(0, 1)
K1 = round(0, 1)
U1 = ((200 / 3) / 46) * 100 * float(var2.get())
MOP1 = (25 / 60) * 100 * float(var2.get())
Ni2 = round(U1, 1)
P2 = round(0, 1)
K2 = round(MOP1, 1)
Ur = (Ni + Ni1 + Ni2) * Urea
Da = (P + P1 + P2) * DAP_price
Mo = (K + K1 + K2) * MOP_price
T = Ur + Da + Mo
x = c3.get()
HL2 = (d1[x][1]) * float(var2.get())
AL2 = (d1[x][2]) * float(var2.get())
ML2 = (d1[x][3]) * float(var2.get())
Seed2 = (d1[x][4]) * float(var2.get())
FM2 = T
z2 = d1[x].replace(d1[x][5], FM2)
l2.append(z2[5])
Insec2 = (d1[x][6]) * float(var2.get())
Irri2 = (d1[x][7]) * float(var2.get())
CI2 = (d1[x][8]) * float(var2.get())
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:10 10-10-2022


Misc2 = (d1[x][9]) * float(var2.get())
WC2 = (d1[x][10]) * float(var2.get())
FC2 = (d1[x][11]) * float(var2.get())
Total2 = HL2 + AL2 + ML2 + Seed2 + FM2 + Insec2 + Irri2 + CI2 + Misc2 + WC2 + FC2
L2 = Label(root,
text="Cost of Cultivation for Maize:\t\t"+ "Rs\t" + str(round(Total2, 0))).place(x=80, y=450)
else:
msgbox.showinfo(title='FRC', message='Maize Crop Area cannot be Zero or Empty')
def Kharif_Total_paddy():
global l1
x = c3.get()
if (var3.get() != 0.0 and var3.get() != " "):
l1 = []
MOP = (45 / 60) * 100 * float(var3.get())
DAP = (90 / 46) * 100 * float(var3.get())
z = ((18 * DAP) / 100) / float(var3.get())
N1 = 60 - z
N2 = (N1 / 46) * 100 * float(var3.get())
Ni = round(N2, 1)
P = round(DAP, 1)
K = round(MOP, 1)
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:10 10-10-2022

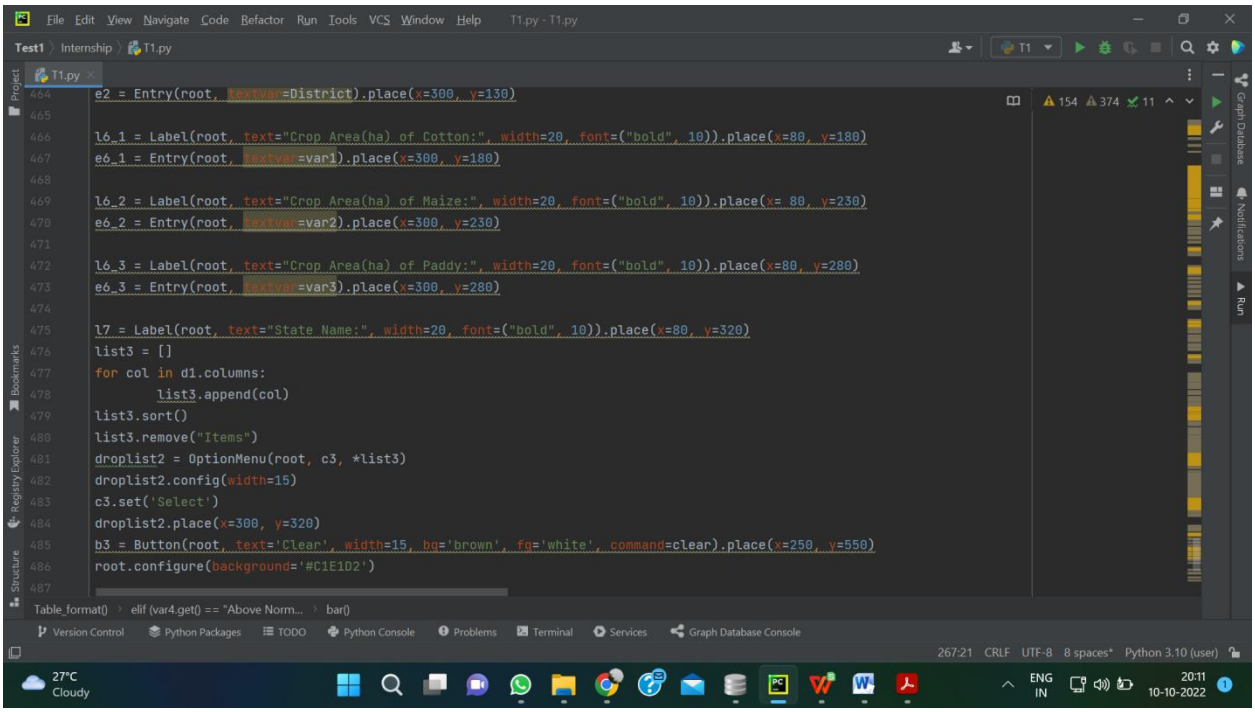

U = (60 / 46) * 100 * float(var3.get())
Ni1 = round(U, 1)
P1 = round(0, 1)
K1 = round(0, 1)
U1 = (60 / 46) * 100 * float(var3.get())
MOP1 = (45 / 60) * 100 * float(var3.get())
Ni2 = round(U1, 1)
P2 = round(0, 1)
K2 = round(MOP1, 1)
Ur = (Ni + Ni1 + Ni2) * Urea
Da = (P + P1 + P2) * DAP_price
Mo = (K + K1 + K2) * MOP_price
T = Ur + Da + Mo
x = c3.get()
HL3 = (d2[x][1]) * float(var3.get())
AL3 = (d2[x][2]) * float(var3.get())
ML3 = (d2[x][3]) * float(var3.get())
Seed3 = (d2[x][4]) * float(var3.get())
FM3 = T
z3 = d2[x].replace(d2[x][5], FM3)
l1.append(z3[5])
Insec3 = (d2[x][6]) * float(var3.get())
Irri3 = (d2[x][7]) * float(var3.get())
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:11 10-10-2022

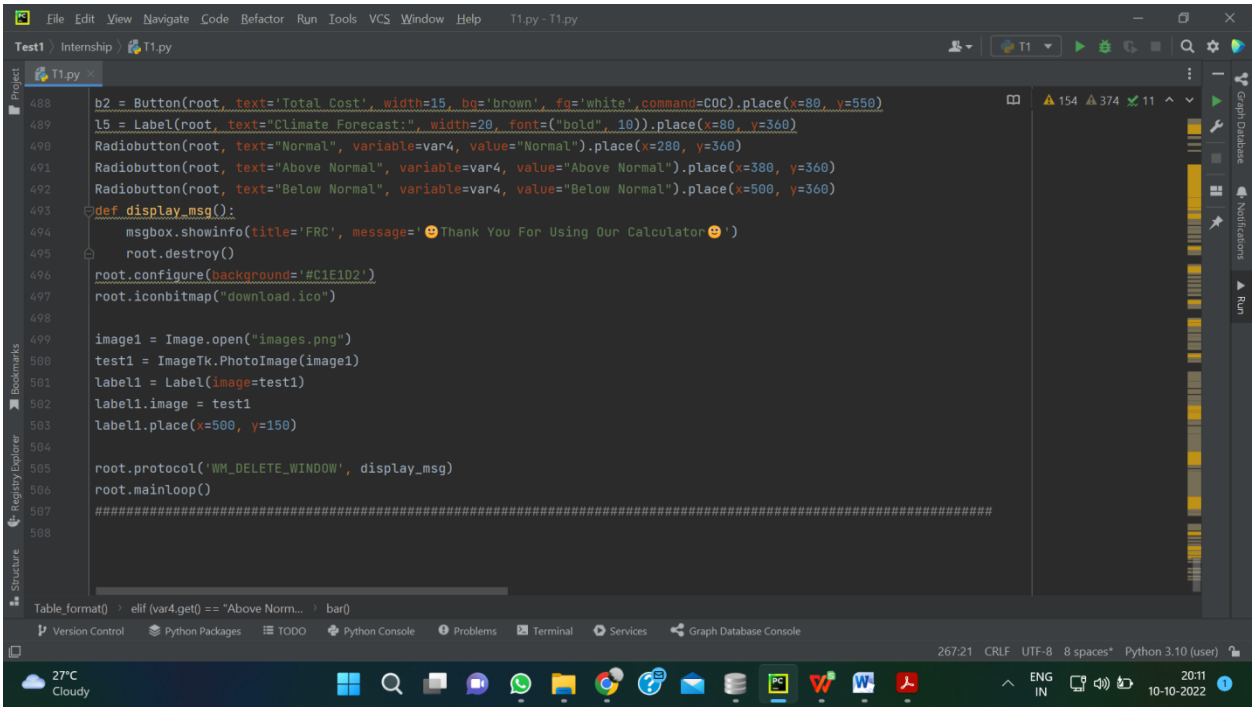


```
                CI3 = (d2[x][8]) * float(var3.get())
                Misc3 = (d2[x][9]) * float(var3.get())
                WC3 = (d2[x][10]) * float(var3.get())
                FC3 = (d2[x][11]) * float(var3.get())
                Total3 = HL3 + AL3 + ML3 + Seed3+ FM3 + Insec3 + Irri3 + CI3 + Misc3 + WC3 + FC3
                L3 = Label(root,
                           text="Cost of Cultivation for Paddy\t\t"+ "Rs\t" + str(round(Total3, 0))).place(x=80, y=500)
        else:
                msgbox.showinfo(title='FRC', message='Paddy Crop Area cannot be Zero or Empty')
# GUI tkinter Part#########################################################################################################################
root = Tk()
c3 = StringVar()
root.geometry('700x800')
root.title("FRC")
Name = StringVar()
District = StringVar()
var1 = DoubleVar()
var2 = DoubleVar()
var3 = DoubleVar()
var4 = StringVar()
Urea=5.52
DAP_price=22.5
MOP_price=17.77
global L1,L2,L3
```

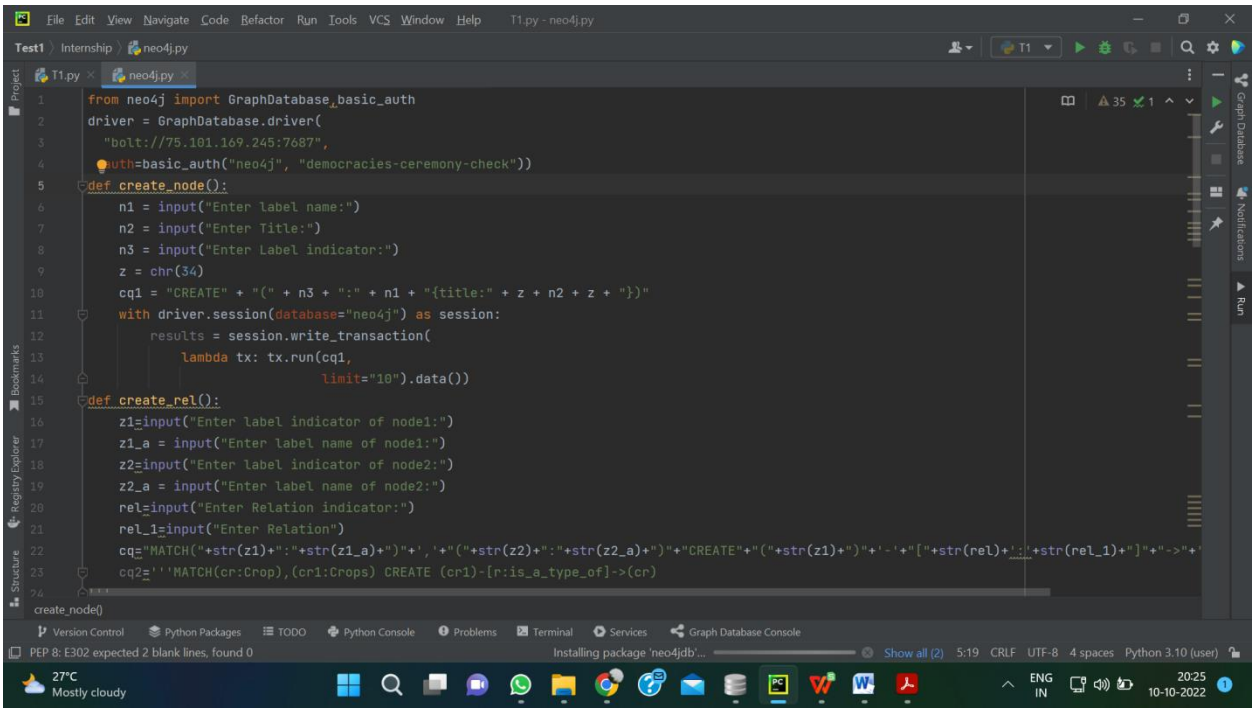


```
global L1,L2,L3
global Total1,Total2,Total3,Z1

def clear():
    Name.set("")
    District.set("")
    var1.set("")
    var2.set("")
    var3.set("")
    c3.set("Select")

def COC():
        Btotal()
        Kharif_Total_maize()
        Kharif_Total_paddy()
        b5 = Button(root, text='Details', width=15, bg='brown', fg='white', command=Table_format).place(x=450, y=450)


l0 = Label(root, text="FERTIGURU", font=("bold", 20), foreground="Red",
           background="#FFFDD0").place(x=240, y=20)
l1 = Label(root, text="Name of the FPO", width=20, font=("bold", 10)).place(x=80, y=80)
e1 = Entry(root, textvar=Name).place(x=300, y=80)
l2 = Label(root, text="District Name", width=20, font=("bold", 10)).place(x=80, y=130)
```

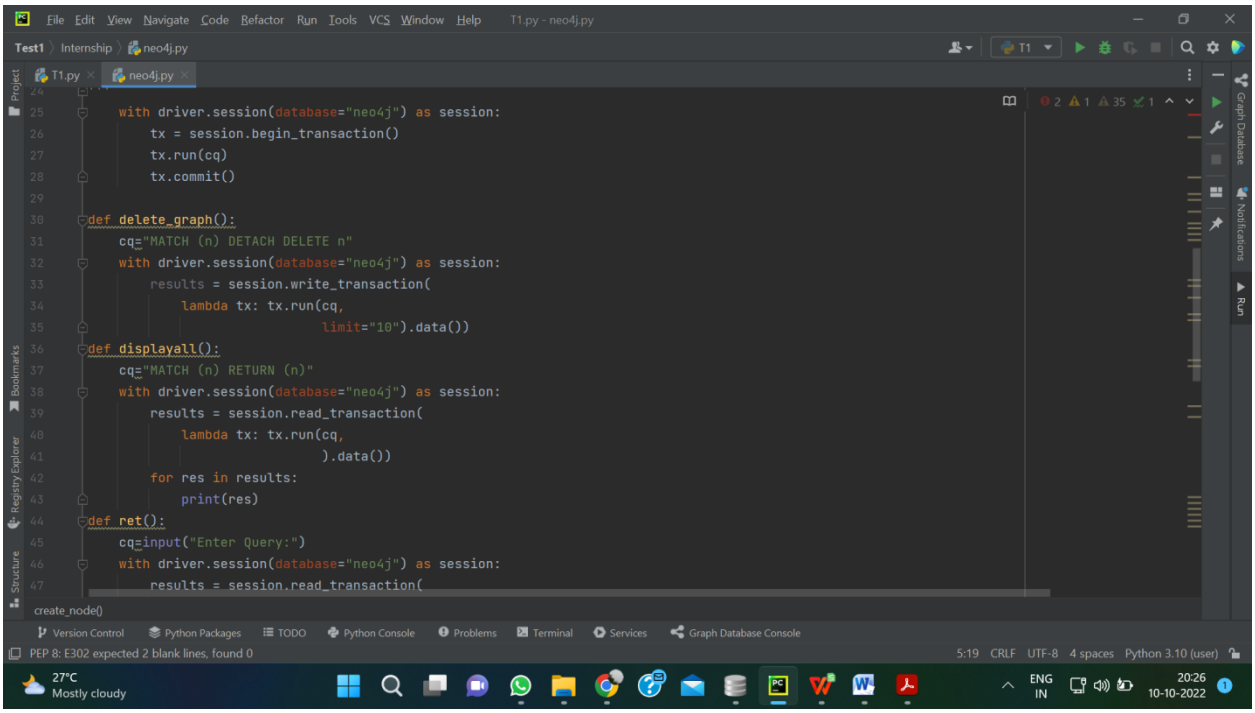

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
e2 = Entry(root, textvar=District).place(x=300, y=130)
l6_1 = Label(root, text="Crop Area(ha) of Cotton:", width=20, font=("bold", 10)).place(x=80, y=180)
e6_1 = Entry(root, textvar=var1).place(x=300, y=180)
l6_2 = Label(root, text="Crop Area(ha) of Maize:", width=20, font=("bold", 10)).place(x= 80, y=230)
e6_2 = Entry(root, textvar=var2).place(x=300, y=230)
l6_3 = Label(root, text="Crop Area(ha) of Paddy:", width=20, font=("bold", 10)).place(x=80, y=280)
e6_3 = Entry(root, textvar=var3).place(x=300, y=280)
l7 = Label(root, text="State Name:", width=20, font=("bold", 10)).place(x=80, y=320)
list3 = []
for col in d1.columns:
list3.append(col)
list3.sort()
list3.remove("Items")
droplist2 = OptionMenu(root, c3, *list3)
droplist2.config(width=15)
c3.set('Select')
droplist2.place(x=300, y=320)
b3 = Button(root, text='Clear', width=15, bg='brown', fg='white', command=clear).place(x=250, y=550)
root.configure(background='#C1E1D2')
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:11 10-10-2022


File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship T1.py
T1.py
b2 = Button(root, text='Total Cost', width=15, bg='brown', fg='white',command=COC).place(x=80, y=550)
l5 = Label(root, text="Climate Forecast:", width=20, font=("bold", 10)).place(x=80, y=360)
Radiobutton(root, text="Normal", variable=var4, value="Normal").place(x=280, y=360)
Radiobutton(root, text="Above Normal", variable=var4, value="Above Normal").place(x=380, y=360)
Radiobutton(root, text="Below Normal", variable=var4, value="Below Normal").place(x=500, y=360)
def display_msg():
msgbox.showinfo(title='FRC', message='Thank You For Using Our Calculator')
root.destroy()
root.configure(background='#C1E1D2')
root.iconbitmap("download.ico")
image1 = Image.open("images.png")
test1 = ImageTk.PhotoImage(image1)
label1 = Label(image=test1)
label1.image = test1
label1.place(x=500, y=150)
root.protocol('WM_DELETE_WINDOW', display_msg)
root.mainloop()
Table_format() elif (var4.get() == "Above Norm... bar()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
27°C Cloudy
ENG IN
20:11 10-10-2022

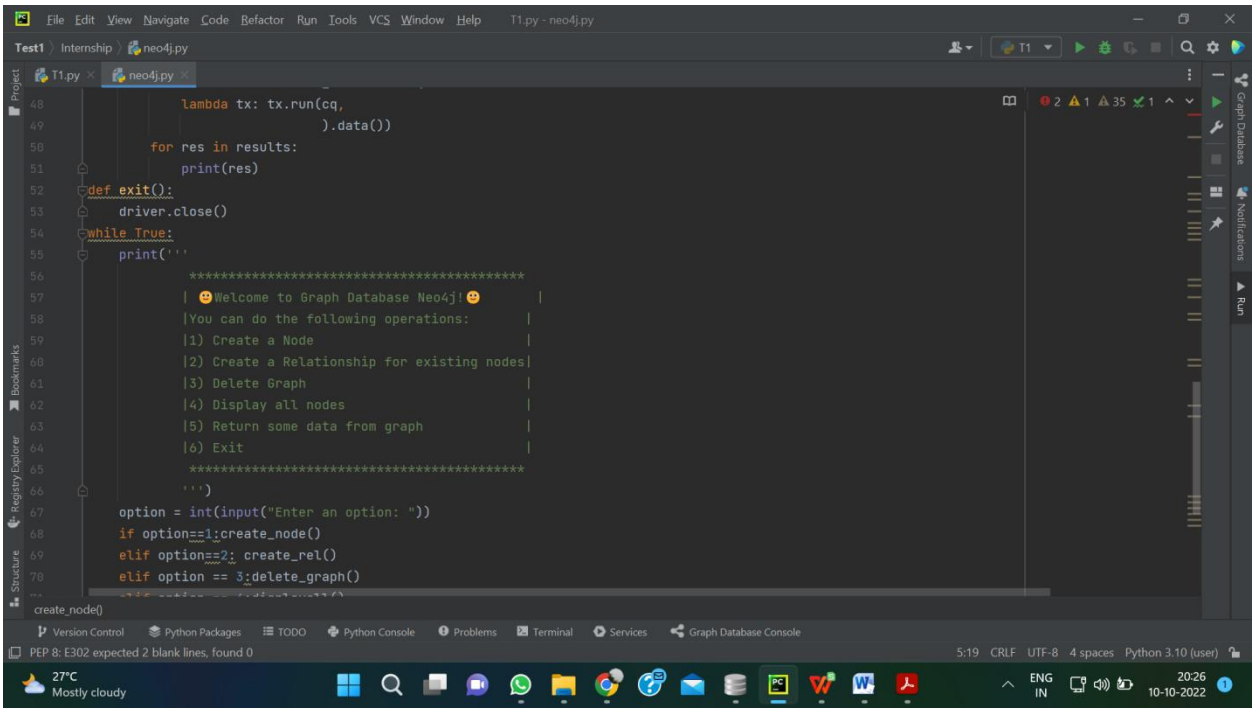
PART-2

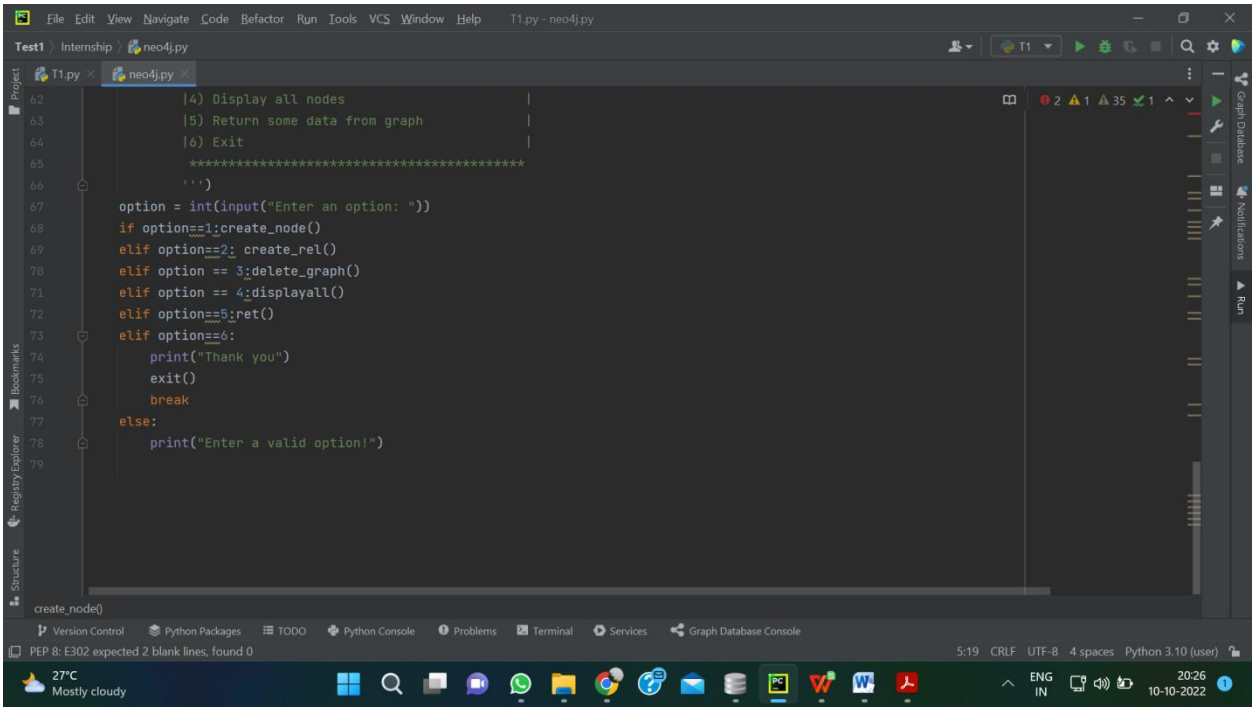


```
from neo4j import GraphDatabase,basic_auth
driver = GraphDatabase.driver(
  "bolt://75.101.169.245:7687",
 auth=basic_auth("neo4j", "democracies-ceremony-check"))
def create_node():
    n1 = input("Enter label name:")
    n2 = input("Enter Title:")
    n3 = input("Enter Label indicator:")
    z = chr(34)
    cq1 = "CREATE" + "(" + n3 + ":" + n1 + "{title:" + z + n2 + z + "})"
    with driver.session(database="neo4j") as session:
        results = session.write_transaction(
            lambda tx: tx.run(cq1,
                              limit="10").data())
def create_rel():
    z1=input("Enter label indicator of node1:")
    z1_a = input("Enter label name of node1:")
    z2=input("Enter label indicator of node2:")
    z2_a = input("Enter label name of node2:")
    rel=input("Enter Relation indicator:")
    rel_1=input("Enter Relation")
    cq="MATCH("+str(z1)+":"+str(z1_a)+")"+','+"("+str(z2)+":"+str(z2_a)+")"+"CREATE"+"("+str(z1)+")"+'-'+"["+str(rel)+':'+str(rel_1)+"]"+"->"+'
    cq2='''MATCH(cr:Crop),(cr1:Crops) CREATE (cr1)-[r:is_a_type_of]->(cr)
'''
```



```
    with driver.session(database="neo4j") as session:
        tx = session.begin_transaction()
        tx.run(cq)
        tx.commit()

def delete_graph():
    cq="MATCH (n) DETACH DELETE n"
    with driver.session(database="neo4j") as session:
        results = session.write_transaction(
            lambda tx: tx.run(cq,
                              limit="10").data())
def displayall():
    cq="MATCH (n) RETURN (n)"
    with driver.session(database="neo4j") as session:
        results = session.read_transaction(
            lambda tx: tx.run(cq,
                              ).data())
        for res in results:
            print(res)
def ret():
    cq=input("Enter Query:")
    with driver.session(database="neo4j") as session:
        results = session.read_transaction(
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - neo4j.py
Test1 Internship neo4j.py
T1.py neo4j.py
lambda tx: tx.run(cq,
).data())
for res in results:
print(res)
def exit():
driver.close()
while True:
print('''
****
| Welcome to Graph Database Neo4j! |
|You can do the following operations: |
|1) Create a Node |
|2) Create a Relationship for existing nodes|
|3) Delete Graph |
|4) Display all nodes |
|5) Return some data from graph |
|6) Exit |
****
''')
option = int(input("Enter an option: "))
if option==1:create_node()
elif option==2: create_rel()
elif option == 3:delete_graph()
create_node()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
PEP 8: E302 expected 2 blank lines, found 0
5:19 CRLF UTF-8 4 spaces Python 3.10 (user)
27°C Mostly cloudy
ENG IN
20:26 10-10-2022


File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - neo4j.py
Test1 Internship neo4j.py
T1.py neo4j.py
|4) Display all nodes |
|5) Return some data from graph |
|6) Exit |
****
''')
option = int(input("Enter an option: "))
if option==1:create_node()
elif option==2: create_rel()
elif option == 3:delete_graph()
elif option == 4:displayall()
elif option==5:ret()
elif option==6:
print("Thank you")
exit()
break
else:
print("Enter a valid option!")
create_node()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
PEP 8: E302 expected 2 blank lines, found 0
5:19 CRLF UTF-8 4 spaces Python 3.10 (user)
27°C Mostly cloudy
ENG IN
20:26 10-10-2022

8. RESULTS

PART-1

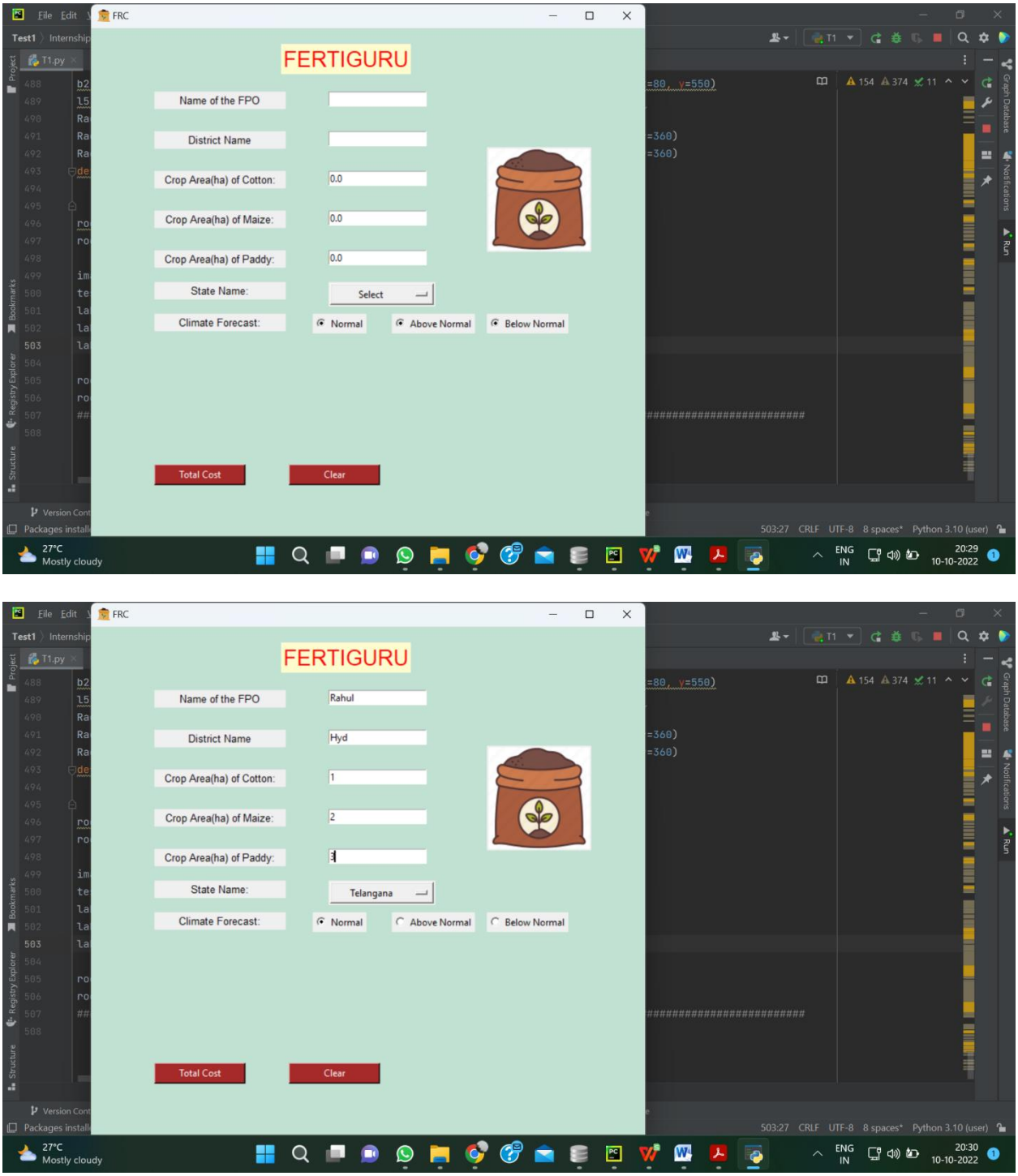
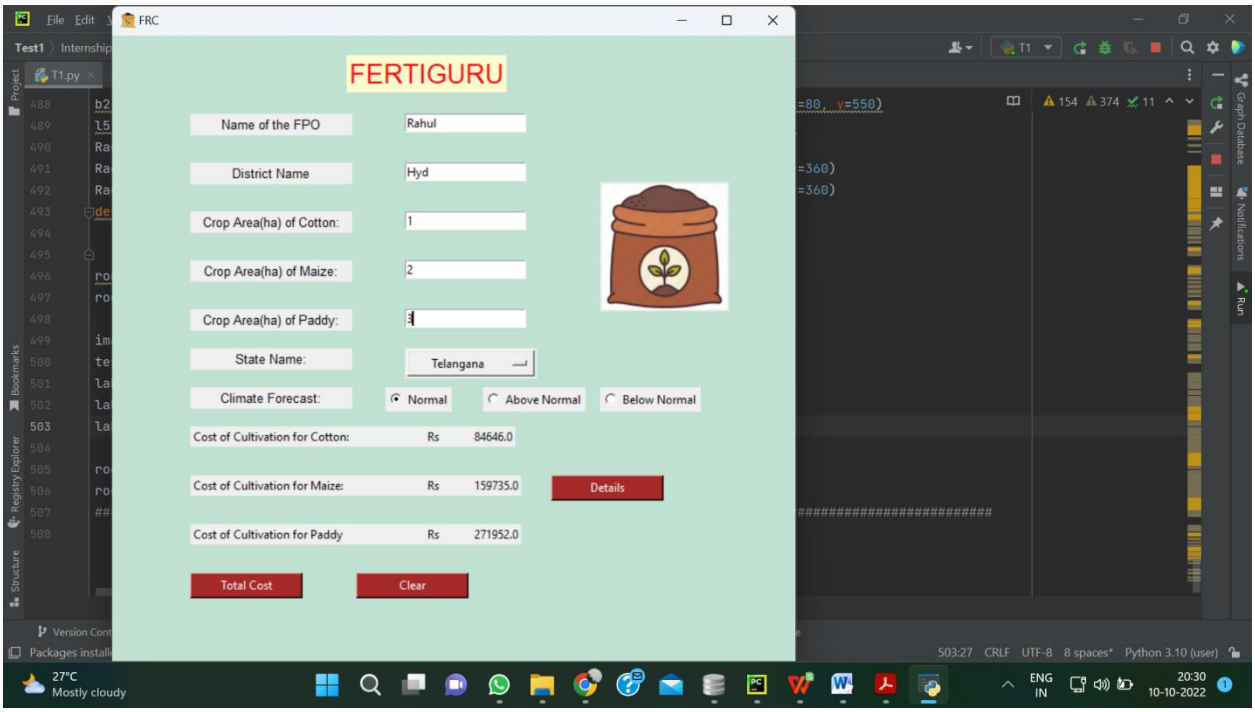


Fig 2. Fertilizer calculator



GREEN SHOWS PROFIT…

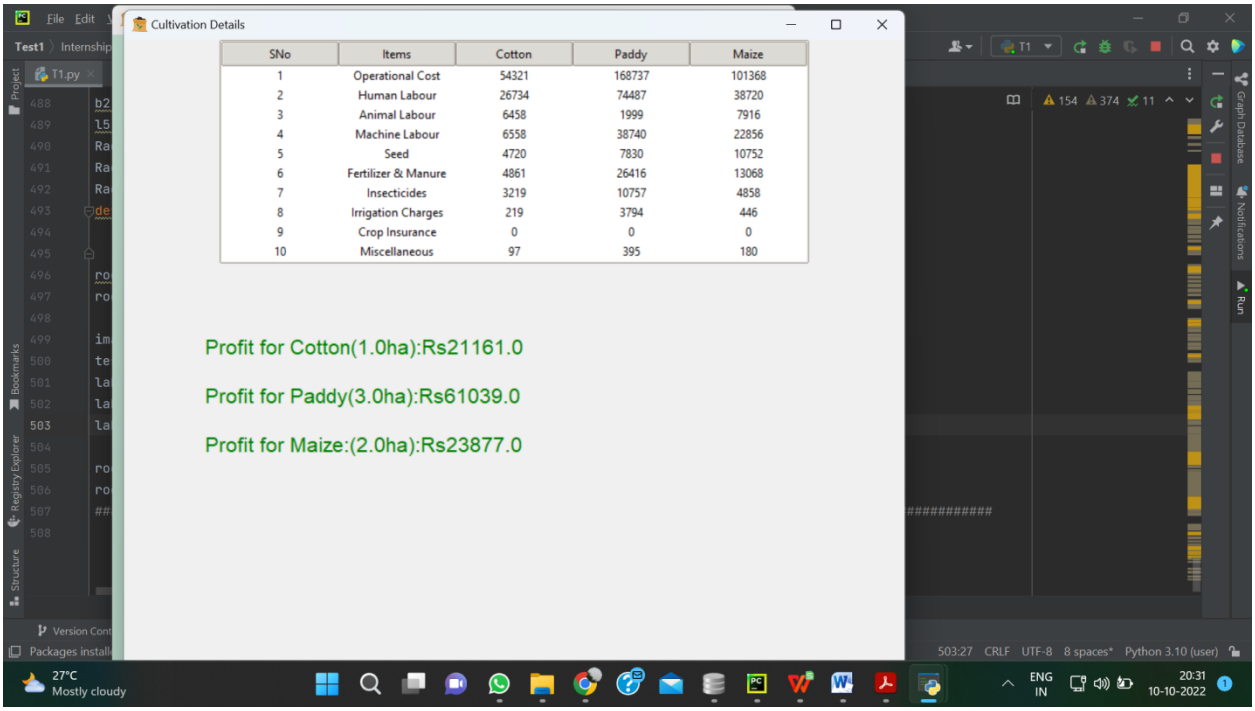


Fig 3 Visualization tool showing cost of cultivation and Net Returns

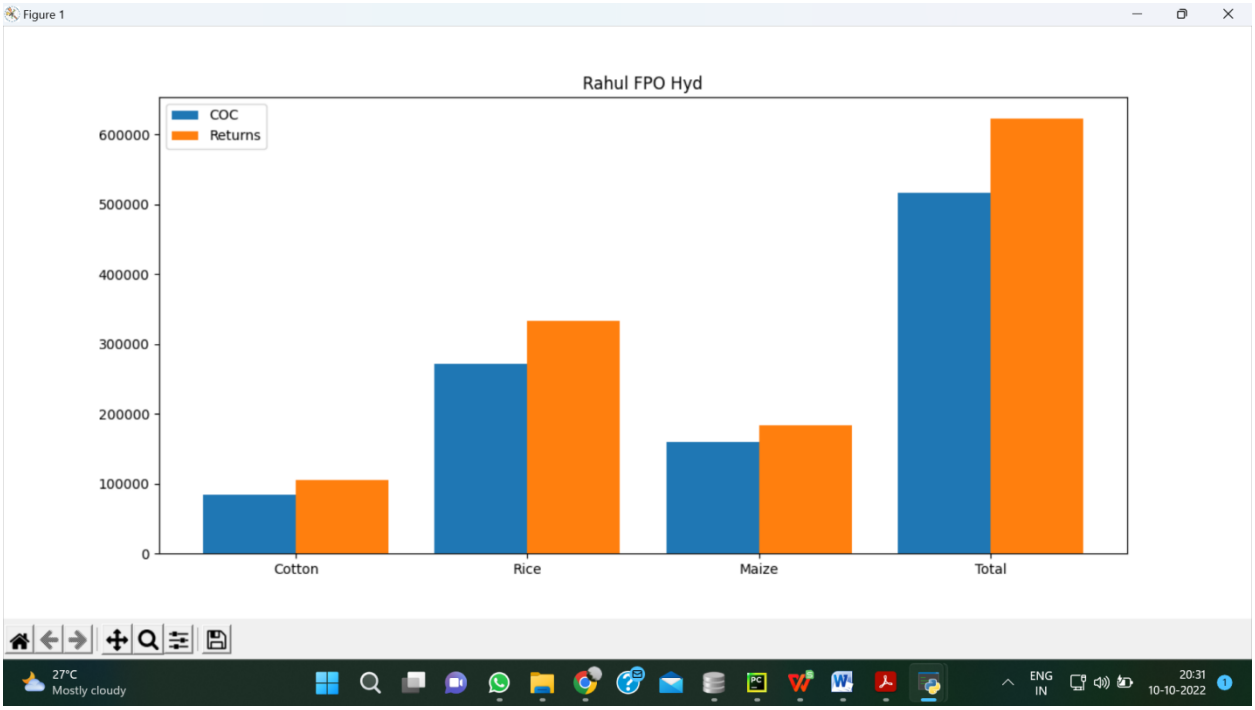
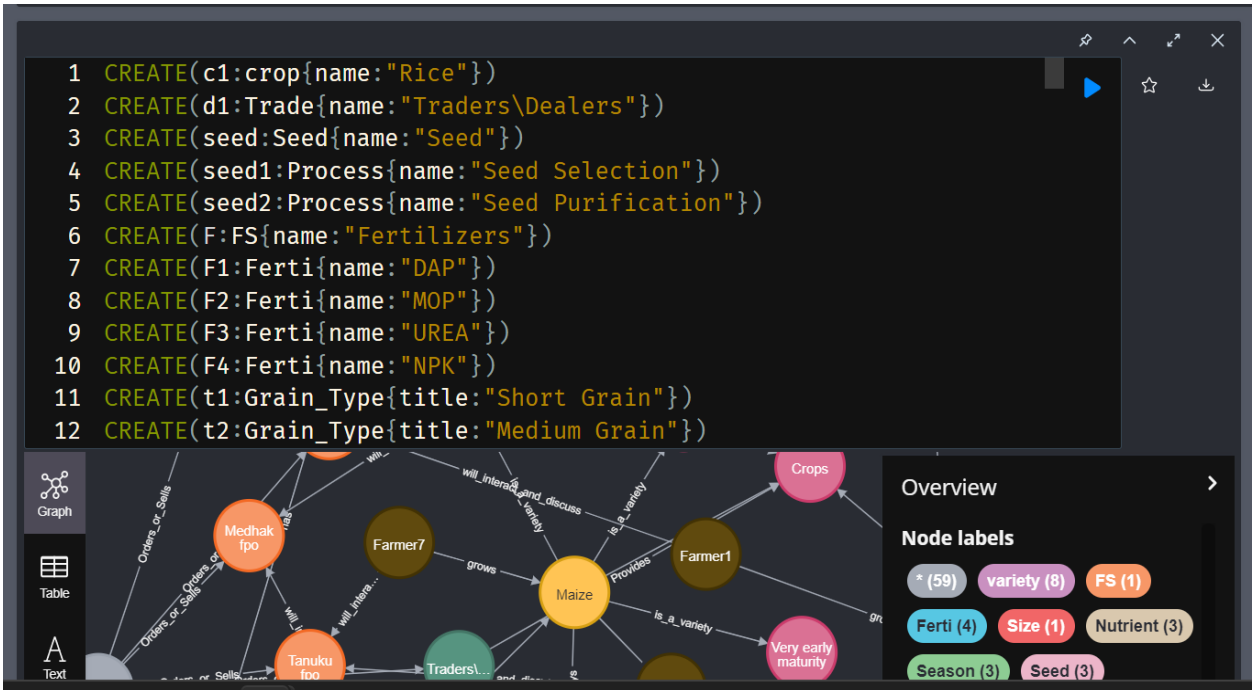


Fig 4. Visualization tool showing cost of cultivation and net returns for overall farm

PART-2



9. CONCLUSION & FUTURE SCOPE

The Yield and profit Prediction can accurately provide the values which helps farmers to grow these three crops and allocate the land accurately to maximize the profit. This tool will also help the FPOS to use the fertilizers calculate which can help in application of correct amounts to prevent the over usage or less usage. Excess application fertilizers will result in environmental pollution and also increases cost of cultivation

The tool also predicts the profit for the crops cultivated. In future we can integrate with other IT tools currently in operation such as sending message to farmers on when to apply fertilizers and other details. The tool can be scalable to several location by linking it with AI and big data tools which automatically gather information from the IMD(New Delhi) and other Agro sites. This will be an out to out interactive tool which will be beneficial with some more future Technology Developments

10. REFERENCES

[1] Raju KV, Kumar R, Vikraman S, Moses Shyam D, Srikanth R, Kumara Charyulu D and Wani SP. 2017. Farmer Producer Organization in Andhra Pradesh: A Scoping Study. Rythu Kosam Project. Research Report IDC-16. Patancheru 502 324. Telangana, India: International Crops Research Institute for the Semi-Arid Tropics. 160 pp. ISBN 978-92-9066-592-2

[2] Rao KPC, Dakshina Murthy K, Dhulipala R, Bhagyashree SD, Gupta MD, Sreepada S, Whitbread AM. 2019. Delivering climate risk information to farmers at scale: the Intelligent Agricultural Systems Advisory Tool (ISAT). CCAFS Working Paper no. 243. Wageningen, the Netherlands: CGIAR Research Program on Climate Change, Agriculture and Food Security (CCAFS). Available online at: www.ccafs.cgiar.org

[3] Srikanth R and Alexaander K 2018. Automated plant disease diagnosis using innovative android App (Plantix) for farmers in Indian state of Andhra Pradesh. International Congress for Plant Pathologists: Boston, USA

[4] Indev consultancy 2022. Automation for Farmer Producer Organization, accessed 10 October 2022, https://www.indevconsultancy.com/

[5] Kehti Buddy 2022, Software to increase revenue from farm business, accessed 10 October 2022, https://khetibuddy.com/

[6] FarmERP, 2022. Digital Agriculture with the Core of Sustainability. accessed 10 October 2022. https://www.farmerp.com/