***Advanced Database System Lab***

**Assignment no. 9**

**PRN:** 2020BTECS00005

**Name:** Sanket Shivaji Jadhav

* **Title:** Install & deploy Cloud Databases on Windows.

* **Aim:** Install & deploy MongoDB & CassandraDB on windows. Create a python GUI for CRUD operations.

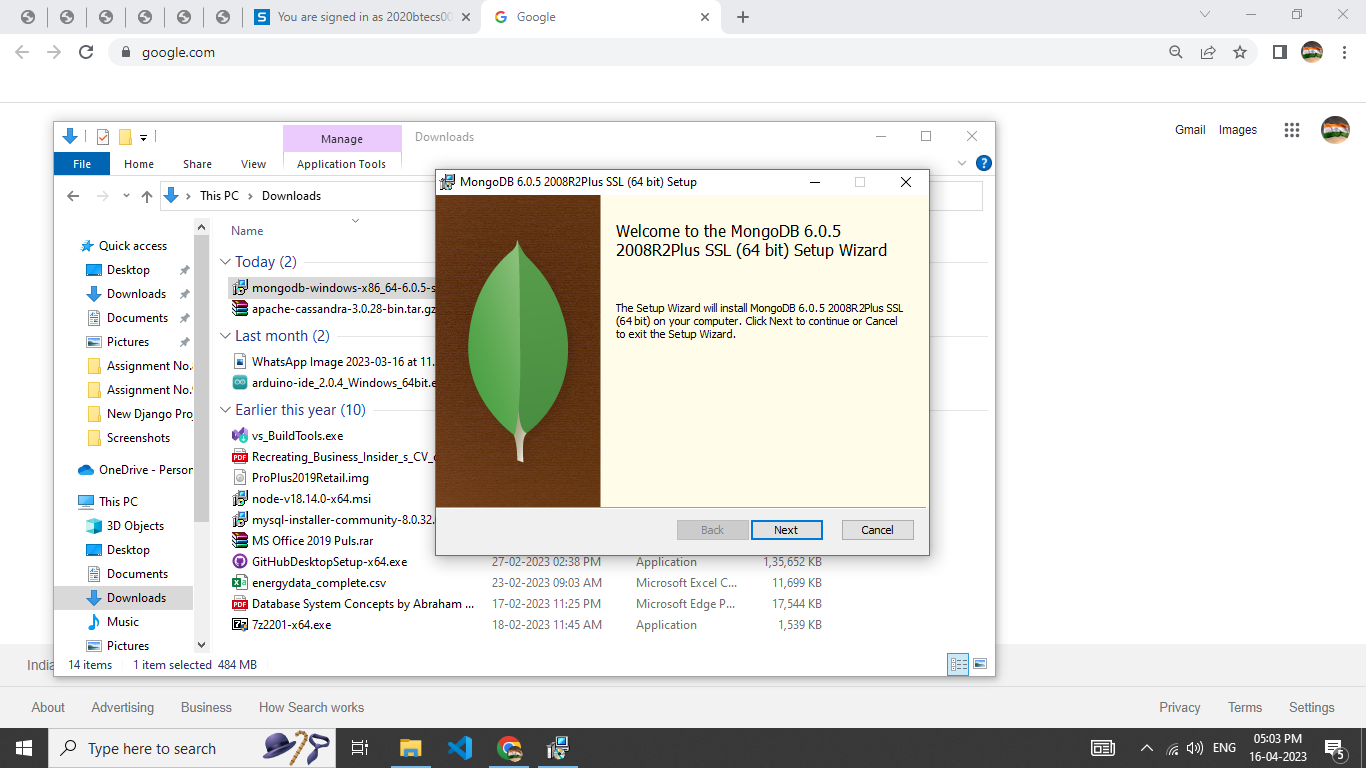
* **Installation:**

**1. MongoDB:**

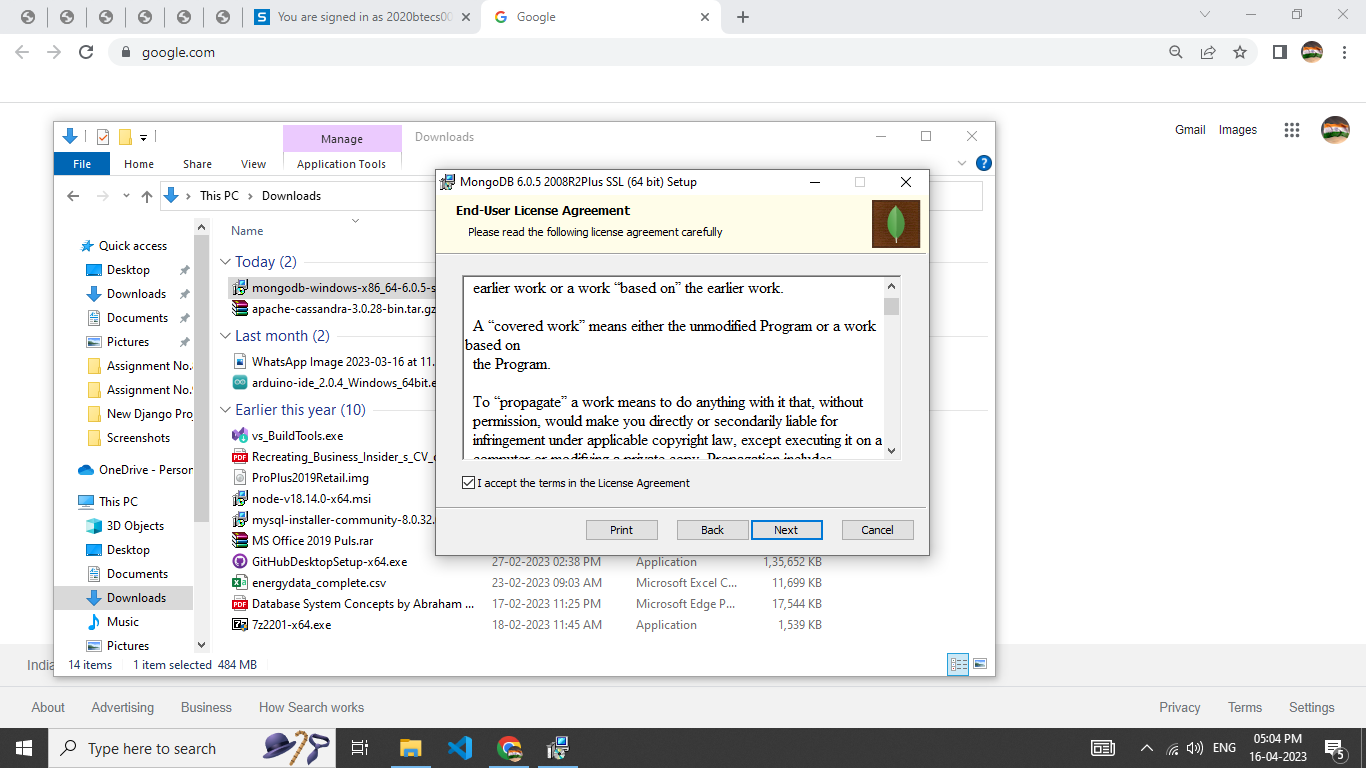
**Step 1.** Download MongoDB Community Server here.

**Step 2.** Once download is complete open the msi file.

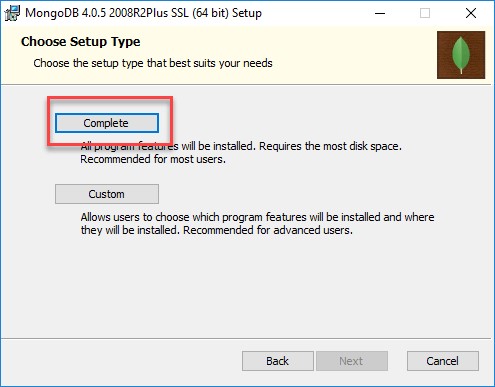
Click Next in the start up screen.



**Step 3.** Accept the End-User License Agreement.



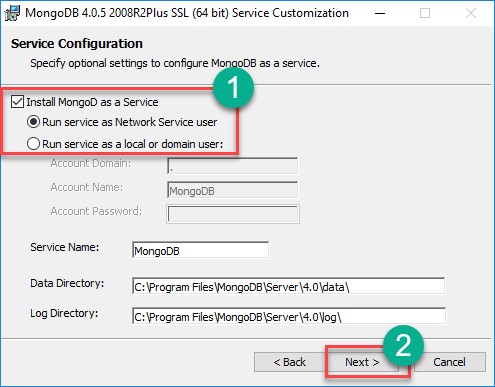
**Step 4.** Click on the “complete” button to install all of the components.



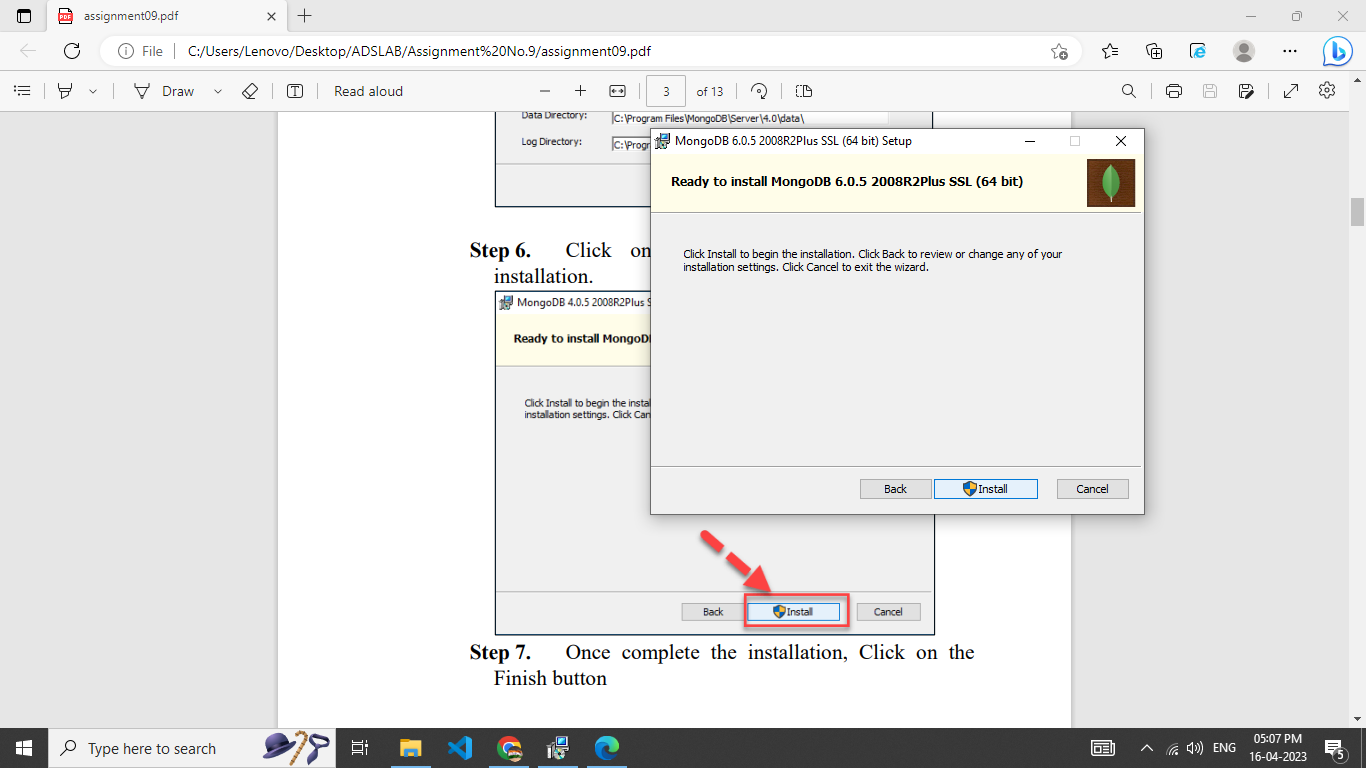
**Step 5.** Select “Run service as Network Service user”.

make a note of the data directory, we’ll need this later.

Click Next.

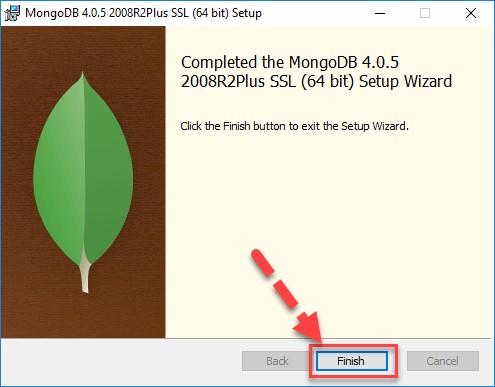


**Step 6.** Click on the Install button to start the installation.



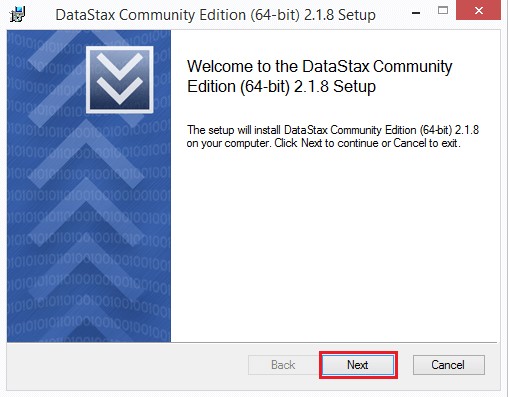
**Step 7.** Once complete the installation, Click on the

Finish button

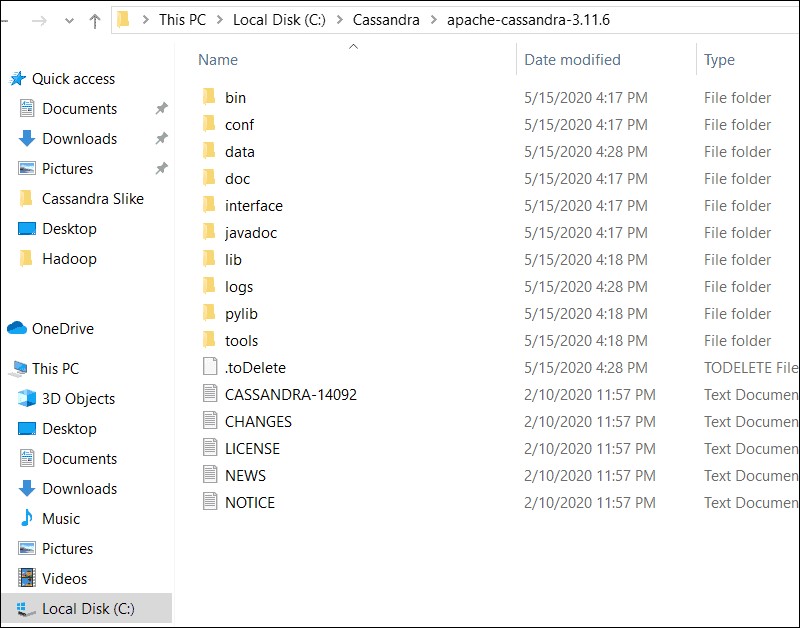


**2. CassandraDB:**

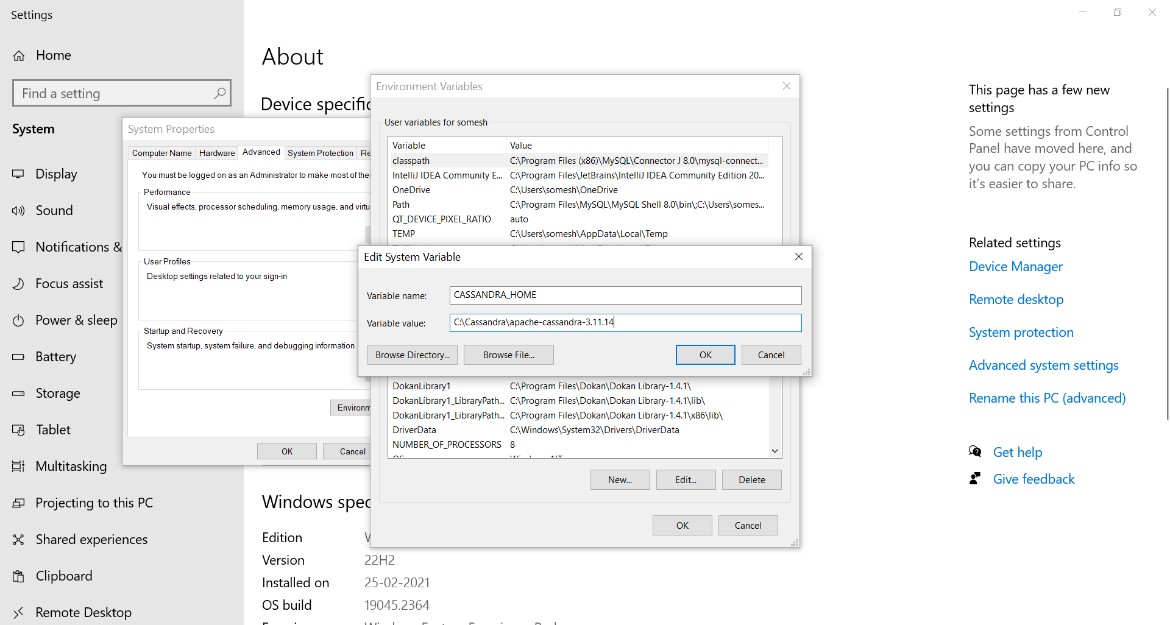
**Step 1.** Download setup from [Apache Downloads](https://www.apache.org/dyn/closer.lua/cassandra/4.0.8/apache-cassandra-4.0.8-bin.tar.gz) & extract it.



**Step 2.** Unzip the folder, and place the content in the C:Cassandraapache-cassandra-3.11.6 folder.



**Step 3.** Configure the environment variables.



**Step 4.** Start Cassandra from cmd.

**Python GUI Application:**

from tkinter import \*

from tkinter import ttk

from tkinter import simpledialog

import tkinter, tkinter.messagebox

from pymongo import MongoClient

from pymongo.server\_api import ServerApi

from dotenv import load\_dotenv

import os

# Loading the data from .env file

load\_dotenv()

# Getting the .env variables

MONGO\_USER = os.getenv("MONGO\_USER")

MONGO\_PASS = os.getenv("MONGO\_PASS")

MONGO\_DB\_NAME = os.getenv("MONGO\_DB\_NAME")

MONGO\_COLLECTION = os.getenv("MONGO\_COLLECTION")

# Connecting to DB

client = MongoClient(f"mongodb+srv://{MONGO\_USER}:{MONGO\_PASS}@adslab.jtazlad.mongodb.net/test", server\_api=ServerApi('1'))

db = client[MONGO\_DB\_NAME] # Select DB

collection = db[MONGO\_COLLECTION] # Select collection

# Initializing Window

window = Tk()

window.title("MongoDB Database Connectivity") # Title of window

window.geometry('900x900') # Size of window (width X height)

window.configure(background = "white"); # Background color of window

window.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

usr\_name = Label(window ,text = f"Connected to Cloud MongoDB as: Sanket", background = "white").grid(row = 0, column = 1, pady=20)

# CRUD Functions

# 1. View

def view\_tb():

    newWindow = Toplevel(window)

    newWindow.title("VIEW Table")

    newWindow.geometry('1500x900')

    newWindow.configure(background = "white"); # Background color of window

    newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

    Label(newWindow ,text = f"Viewing Collection - Assignment09", background = "white").grid(row = 0, column = 0, padx=10, pady=10)

    # Getting all column names from table

    coll\_keys = collection.find\_one()

    columns = [a for a in coll\_keys]

    tree = ttk.Treeview(newWindow, height=20, columns=columns, show='headings')

    tree.grid(row=1, column=0, sticky='news', padx=10, pady=10)

    # setup columns attributes

    for col in columns:

        tree.heading(col, text=col)

        tree.column(col, width=100, anchor=tkinter.CENTER)

    # populate data to treeview

    all\_data = collection.find({})

    data\_list = []

    for a in all\_data:

        data\_list.append(tuple(a.values()))

    for d in data\_list:

        tree.insert('', 'end', value=d)

    # scrollbar

    sb = tkinter.Scrollbar(newWindow, orient=tkinter.VERTICAL, command=tree.yview)

    sb.grid(row=1, column=1, sticky='ns', padx=0, pady=10)

    tree.config(yscrollcommand=sb.set)

    sbx = tkinter.Scrollbar(newWindow, orient=tkinter.HORIZONTAL, command=tree.xview)

    sbx.grid(row=2, column=0, sticky='ew', padx=10, pady=0)

    tree.config(xscrollcommand=sbx.set)

# 2. Insert

def insert\_tb():

    newWindow = Toplevel(window)

    newWindow.title("INSERT into Table")

    newWindow.geometry('900x900')

    newWindow.configure(background = "white"); # Background color of window

    newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

    Label(newWindow ,text = f"Insert values in collection: Assignment09", background = "white").grid(row = 0, column = 0, padx=10, pady=10)

    # Getting columns names

    coll\_keys = collection.find\_one()

    columns = [a for a in coll\_keys]

    columns.pop(0) # Removing the \_id field (entered automatically)

    ent\_ref = [] # For storing the Entry references

    # Populating Labels and Entries

    for ind, nm in enumerate(columns):

        Label(newWindow ,text = nm, background = "white").grid(row = ind+1, column = 0, padx=10, pady=10)

        ent = Entry(newWindow)

        ent.grid(row = ind+1,column = 1)

        ent\_ref.append(ent)

    def insert\_val():

        val = []

        is\_empty = False

        # Getting value from each entry field

        for r in ent\_ref:

            if len(r.get()) > 0:

                val.append(r.get())

            else:

                tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                is\_empty = True

                break

        # Checking if all fields are filled, before inserting

        if not is\_empty:

            v = []

            # Typecasting values (int, float & string)

            for x in val:

                try:

                    v.append(int(x))

                except ValueError:

                    try:

                        v.append(float(x))

                    except ValueError:

                        v.append(x)

            doc\_obj = dict(zip(columns, v))

            # Inserting values

            try:

                collection.insert\_one(doc\_obj)

                for r in ent\_ref:

                    r.delete(0, END)

                tkinter.messagebox.showinfo("SUCCESS", "Values inserted into Collection successfully!")

            except Exception as e:

                tkinter.messagebox.showerror("ERROR", e)

    Button(newWindow, text="Insert Values", command=insert\_val, background="green", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

# 3. Update

def update\_tb():

    try:

        id = simpledialog.askinteger(title="UPDATE", prompt="Enter the PRN to be updated: ")

        if id is not None:

            query={"PRN":{"$eq":id}}

            present\_data = collection.find\_one(query)

            if present\_data is None:

                tkinter.messagebox.showerror("ERROR", "No record was found with the given PRN !")

            else:

                newWindow = Toplevel(window)

                newWindow.title("UPDATE Table")

                newWindow.geometry('900x900')

                newWindow.configure(background = "white"); # Background color of window

                newWindow.option\_add("\*Font", "Times 16") # Setting the font-family & font-size

                Label(newWindow ,text = f"Update values in collection: Assignment09", background = "white").grid(row = 0, column = 0, padx=10, pady=10)

                coll\_keys = collection.find\_one()

                columns = [a for a in coll\_keys]

                columns.pop(0) # Removing the \_id field (entered automatically)

                ent\_ref = []

                val = []

                for k, v in present\_data.items():

                    val.append(str(v))

                val.pop(0) # Removing ObjectId

                for ind, nm in enumerate(columns):

                    Label(newWindow ,text = nm, background = "white").grid(row = ind+1, column = 0, padx=10, pady=10)

                    ent = Entry(newWindow)

                    ent.grid(row = ind+1,column = 1)

                    ent.insert(0, val[ind])

                    ent\_ref.append(ent)

                def update\_val():

                    upd\_val = []

                    is\_empty = False

                    for r in ent\_ref:

                        if len(r.get()) > 0:

                            upd\_val.append(r.get())

                        else:

                            tkinter.messagebox.showerror("ERROR", "All the fields are required!")

                            is\_empty = True

                            break

                    if not is\_empty:

                        v = []

                        for x in upd\_val:

                            try:

                                v.append(int(x))

                            except ValueError:

                                try:

                                    v.append(float(x))

                                except ValueError:

                                    v.append(x)

                        try:

                            doc\_obj = dict(zip(columns, v))

                            new\_data = {"$set":doc\_obj}

                            collection.update\_one(present\_data, new\_data)

                            newWindow.destroy()

                            tkinter.messagebox.showinfo("SUCCESS", "Values updated successfully!")

                        except Exception as e:

                            tkinter.messagebox.showerror("ERROR", e)

                Button(newWindow, text="Update Values", command=update\_val, background="blue", foreground="white").grid(row = ind+2, column = 1, pady=20, sticky='ew')

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# 4. Delete

def delete\_tb():

    try:

        id = simpledialog.askinteger(title="DELETE", prompt="Enter the PRN to be deleted: ")

        if id is not None:

            query={"PRN":{"$eq":id}}

            present\_data = collection.find\_one(query)

            if present\_data is None:

                tkinter.messagebox.showerror("ERROR", "Cannot DELETE!\nNo record was found with the given PRN !")

            else:

                collection.delete\_one(query)

                tkinter.messagebox.showinfo("SUCCESS", "Deleted record from Collection successfully!")

    except Exception as e:

        tkinter.messagebox.showerror("ERROR", e)

# CRUD operation buttons

Label(window ,text = "Operations on collection:", background = "white", font='Helvetica 18 bold').grid(row = 3, column = 0, padx=10, pady=60)

view\_btn = Button(window, text="View", command=view\_tb, background="#9629ff", foreground="white", border=3).grid(row = 4, column = 0)

insert\_btn = Button(window, text="Insert", command=insert\_tb, background="green", foreground="white", border=3).grid(row = 4, column = 1, sticky='w', columnspan=1)

update\_btn = Button(window, text="Update", command=update\_tb, background="blue", foreground="white", border=3).grid(row = 4, column = 1, columnspan=2)

delete\_btn = Button(window, text="Delete", command=delete\_tb, background="red", foreground="white", border=3).grid(row = 4, column = 2)

window.mainloop() # window remains until user closes it

client.close() # Closing the connection to database

o **Result:**

