NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

Install & deploy the following cloud databases on windows platform :

A] MongoDB

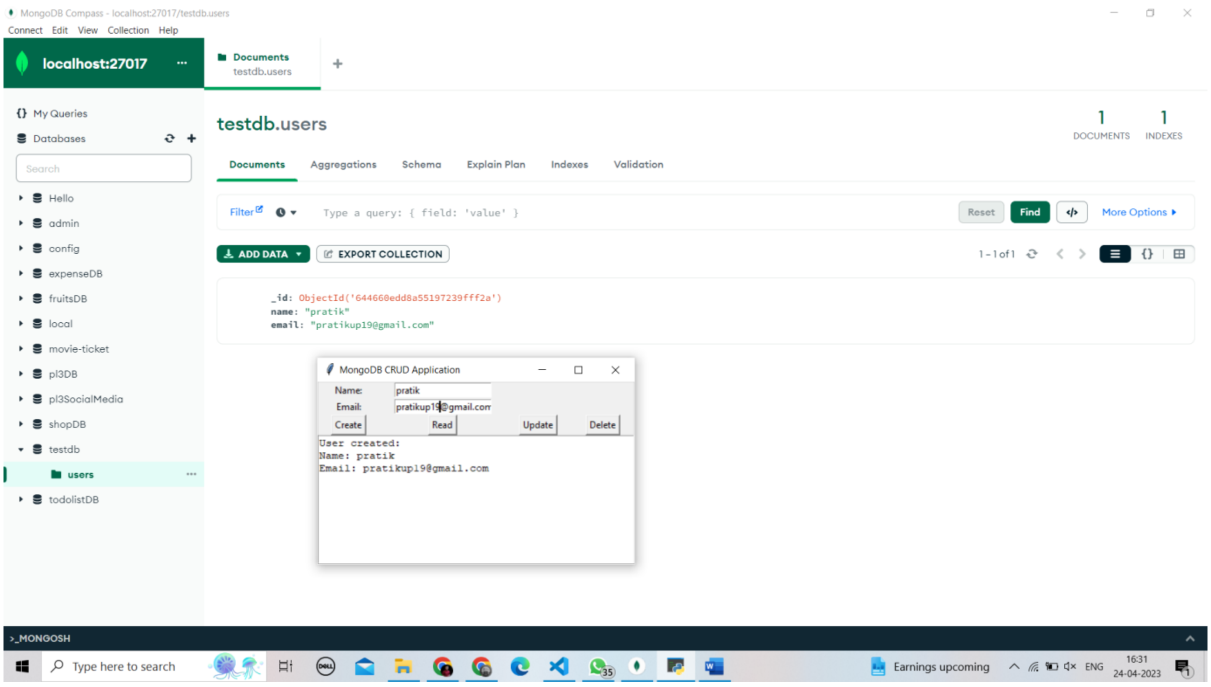
B] CassandraDB

Write Python desktop Application to demonstrate the CRUD operation with

above backend cloud databases. Assume any database.

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

A] MongoDB

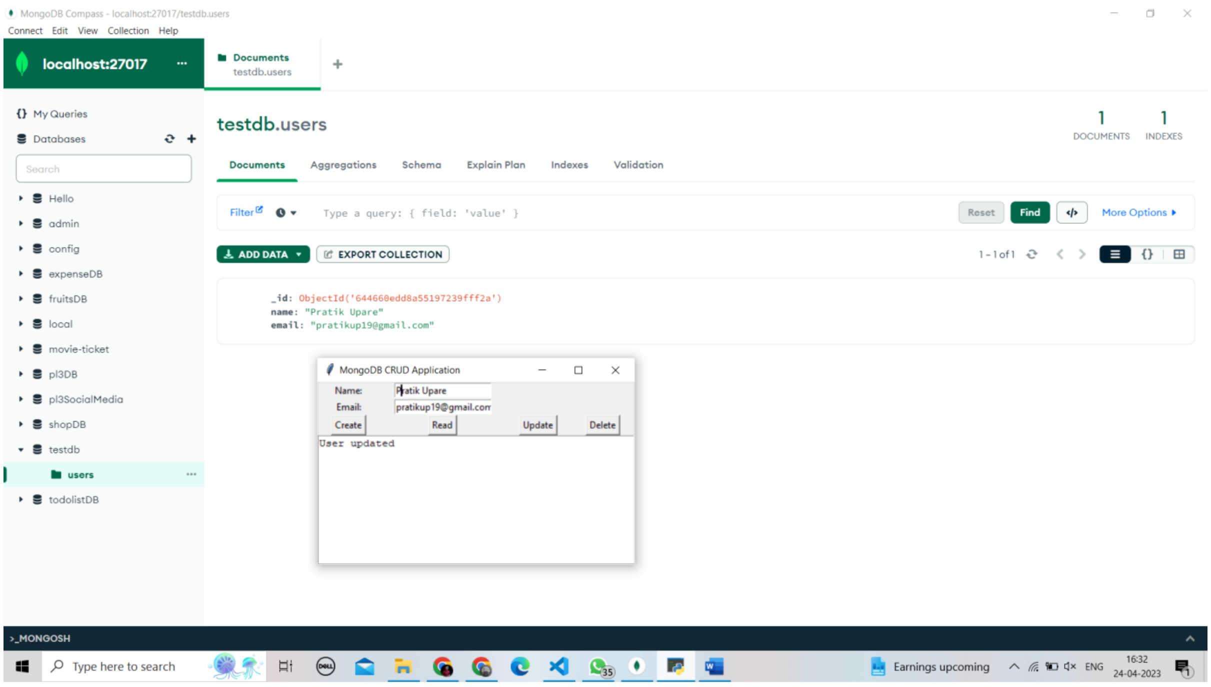


testdb.users

```
_id: ObjectId('644660edd8a55197239fff2a')
name: "pratik"
email: "pratikup19@gmail.com"
```

Name: pratik
Email: pratikup19@gmail.com

```
User created:
Name: pratik
Email: pratikup19@gmail.com
```



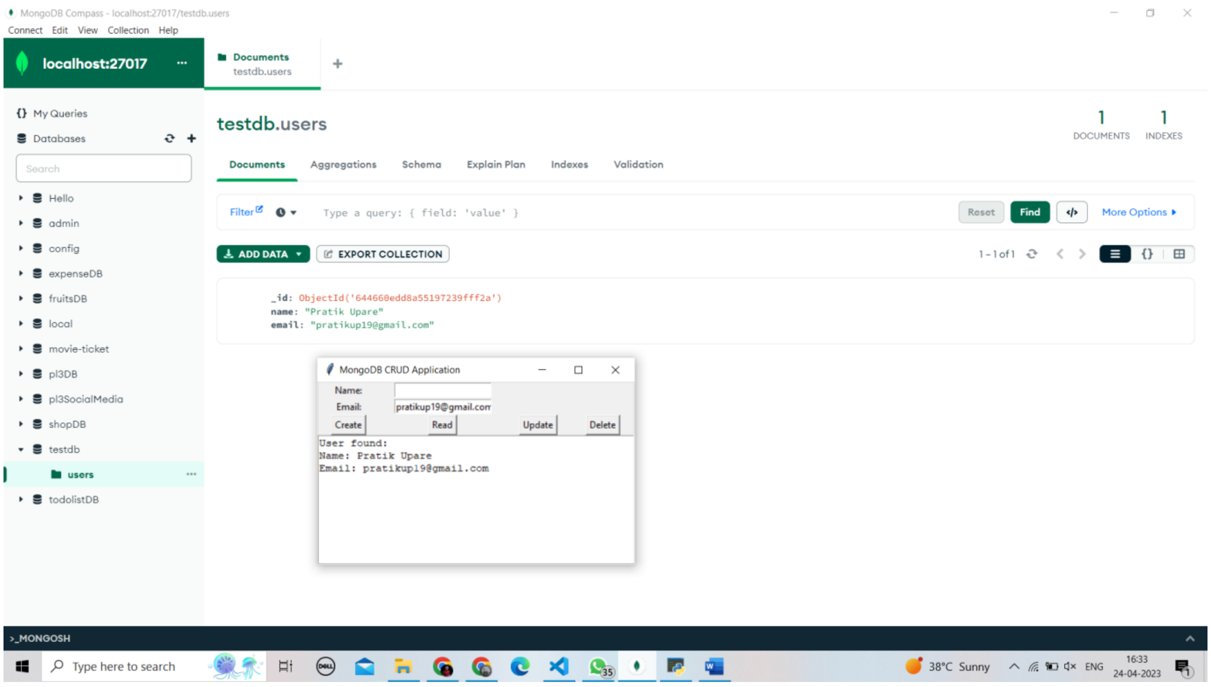
testdb.users

```
_id: ObjectId('644660edd8a55197239fff2a')
name: "Pratik Upare"
email: "pratikup19@gmail.com"
```

Name: Pratik Upare
Email: pratikup19@gmail.com

```
User updated
```

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

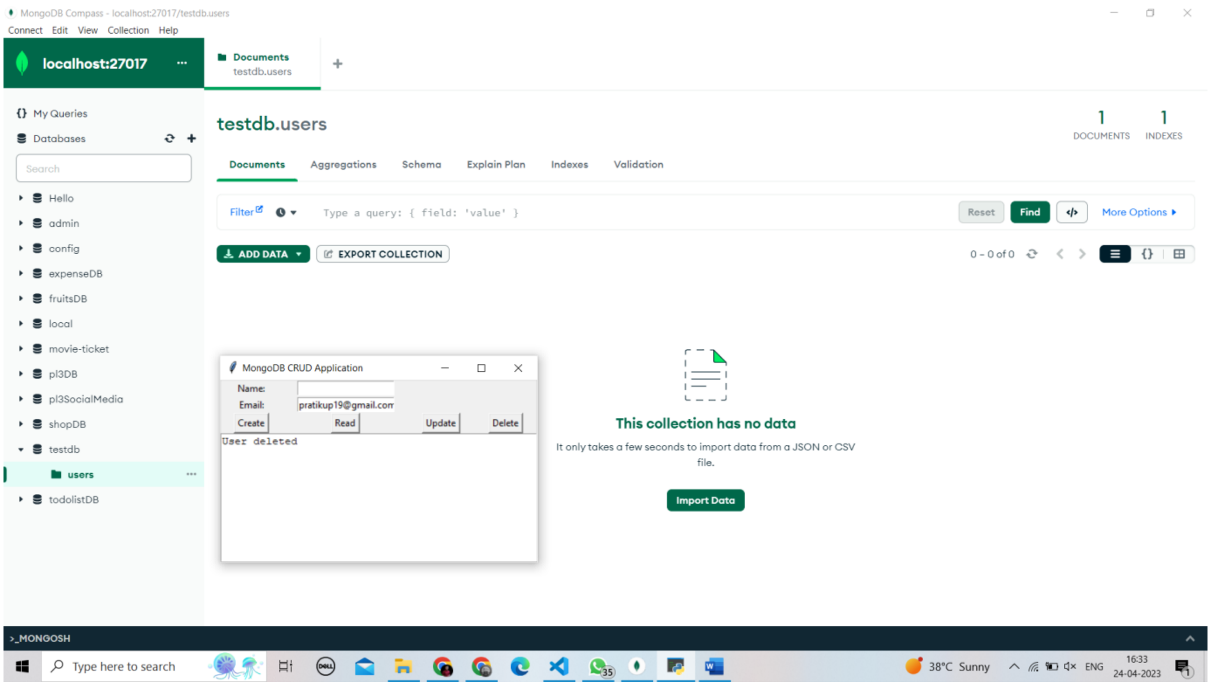


```
_id: ObjectId('644660edd8a55197239fff2a')
name: "Pratik Upare"
email: "pratikup19@gmail.com"
```

MongoDB CRUD Application

Name:
Email: pratikup19@gmail.com

```
User found:
Name: Pratik Upare
Email: pratikup19@gmail.com
```



MongoDB CRUD Application

Name:
Email: pratikup19@gmail.com

```
User deleted
```

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

Code:

```
import pymongo
import tkinter as tk

class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.master.title("MongoDB CRUD Application")
        self.pack()

        # Connect to the MongoDB client and select a database and collection
        self.client = pymongo.MongoClient('mongodb://localhost:27017/')
        self.db = self.client['testdb']
        self.collection = self.db['users']

        # Create labels and entry fields for user details
        self.name_label = tk.Label(self, text="Name:")
        self.name_label.grid(row=0, column=0)
        self.name_entry = tk.Entry(self)
        self.name_entry.grid(row=0, column=1)
        self.email_label = tk.Label(self, text="Email:")
        self.email_label.grid(row=1, column=0)
        self.email_entry = tk.Entry(self)
        self.email_entry.grid(row=1, column=1)

        # Create buttons for CRUD operations
        self.create_button = tk.Button(self, text="Create",
command=self.create_user)
        self.create_button.grid(row=2, column=0)
        self.read_button = tk.Button(self, text="Read",
command=self.read_user)
        self.read_button.grid(row=2, column=1)
        self.update_button = tk.Button(self, text="Update",
command=self.update_user)
        self.update_button.grid(row=2, column=2)
        self.delete_button = tk.Button(self, text="Delete",
command=self.delete_user)
        self.delete_button.grid(row=2, column=3)

        # Create a text box to display user details
        self.text_box = tk.Text(self, height=10, width=50)
        self.text_box.grid(row=3, columnspan=4)

    def create_user(self):
        # Insert a new user
        user = {'name': self.name_entry.get(), 'email':
self.email_entry.get()}
```

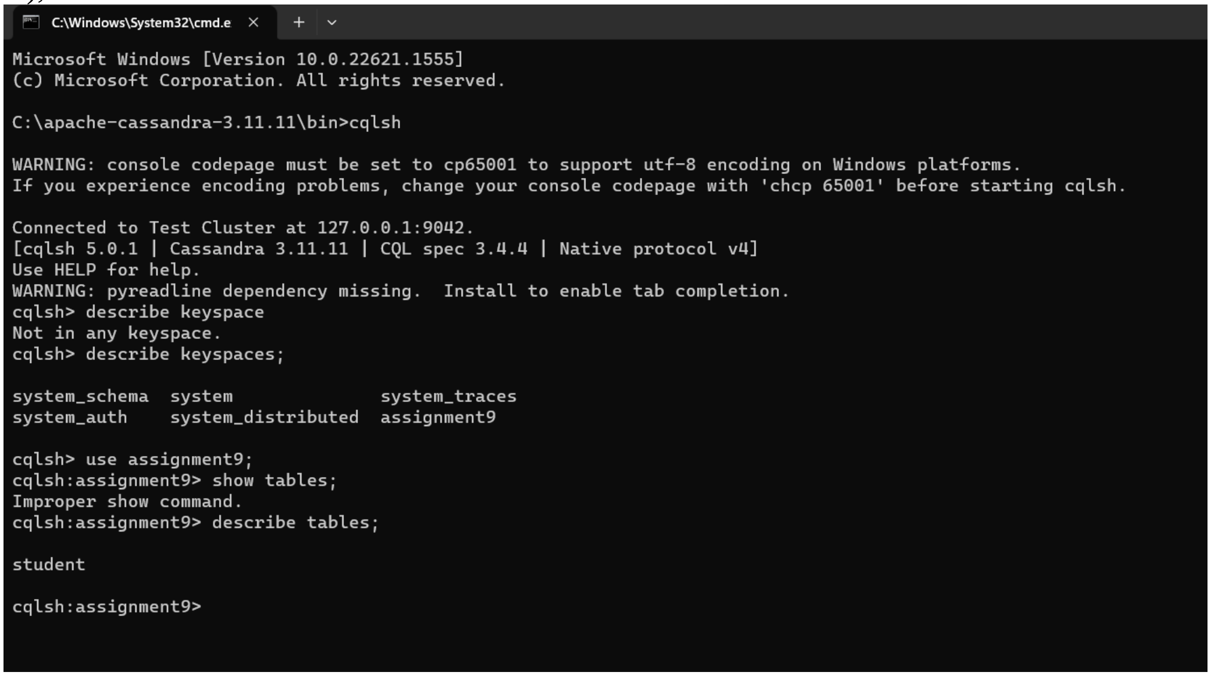
NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

```
        result = self.collection.insert_one(user)
        self.text_box.delete("1.0", tk.END)
        self.text_box.insert(tk.END, "User created:\nName: {}\nEmail:
{}".format(user['name'], user['email']))

    def read_user(self):
        # Find a user by email
        user = self.collection.find_one({'email': self.email_entry.get()})
        self.text_box.delete("1.0", tk.END)
        if user:
            self.text_box.insert(tk.END, "User found:\nName: {}\nEmail:
{}".format(user['name'], user['email']))
        else:
            self.text_box.insert(tk.END, "User not found")

    def update_user(self):
        # Update a user
        result = self.collection.update_one({'email': self.email_entry.get()},
{'$set': {'name': self.name_entry.get()}})
        self.text_box.delete("1.0", tk.END)
        if result.modified_count > 0:
            self.text_box.insert(tk.END, "User updated")
        else:
            self.text_box.insert(tk.END, "User not found")

    def delete_user(self):
        # Delete a user
        result = self.collection.delete_one({'email': self.email_entry.get()})
        self.text_box.delete("1.0", tk.END)
        if result.deleted_count > 0:
            self.text_box.insert(tk.END, "User deleted")
        else:
            self.text_box.insert(tk.END, "User not found")
root = tk.Tk()
app = Application(master=root)
app.mainloop()
```

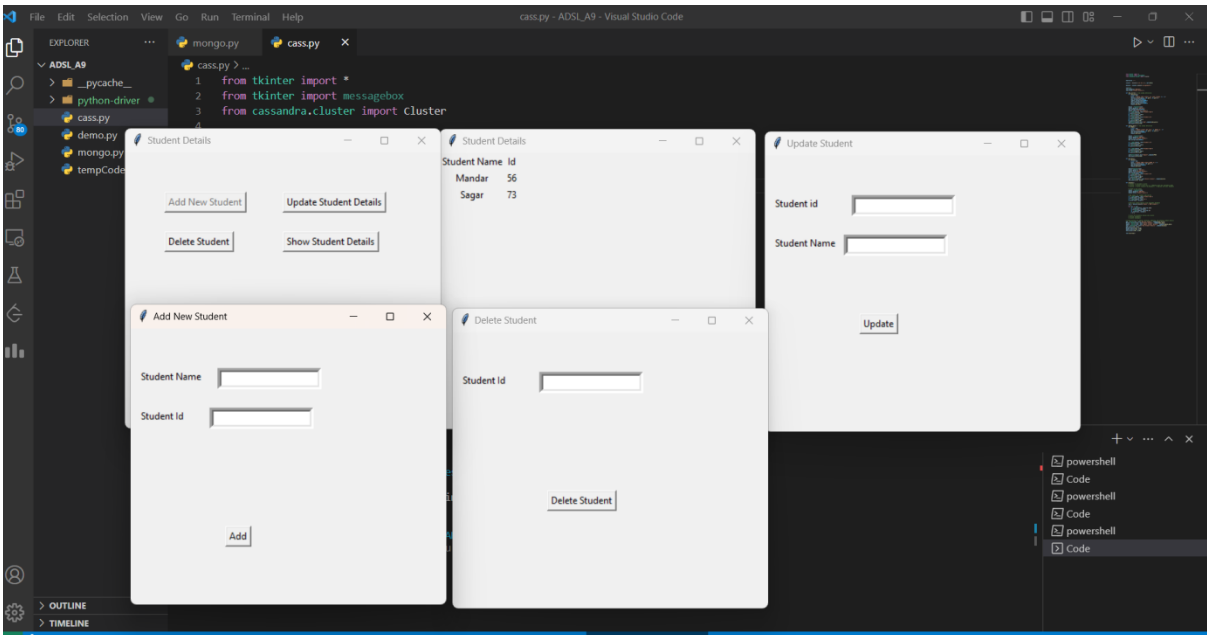
NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

B] CassandraDB

```
CREATE KEYSPACE “assignment9”
WITH replication = {'class': ‘SimpleStrategy
’, 'replication_factor' : ‘1’};

CREATE TABLE student(
  id int PRIMARY KEY,
  name text,
  );
```



```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\apache-cassandra-3.11.11\bin>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.11 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> describe keyspace
Not in any keyspace.
cqlsh> describe keyspaces;

system_schema  system              system_traces
system_auth    system_distributed  assignment9

cqlsh> use assignment9;
cqlsh:assignment9> show tables;
Improper show command.
cqlsh:assignment9> describe tables;

student

cqlsh:assignment9>
```



```
from tkinter import *
from tkinter import messagebox
from cassandra.cluster import Cluster
```

```
Student Name  Id
Mandar        56
Sagar         73
```

```
from tkinter import *
from tkinter import messagebox
from cassandra.cluster import Cluster
```

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

```python
expression = ""

cluster = Cluster(['127.0.0.1'], port=9042)

session = cluster.connect('assignment9')

root=Tk()
root.geometry('400x350')
root.title("Student Details")

def add_course(): # new window definition
    def add_query():
        global root
        query = "INSERT INTO student (id, name) VALUES (%s, %s)"
        session.execute(query, (E2.get(), E1.get()))
        add.config(state=NORMAL)
        update.config(state=NORMAL)
        show.config(state=NORMAL)
        delete.config(state=NORMAL)
        newwin.destroy()

    newwin = Toplevel(root)
    newwin.geometry('400x350')
    add.config(state=DISABLED)
    newwin.title("Add New Student")
    L1 = Label(newwin, text="Student Name")
    L1.place(x=10,y=50)
    E1 = Entry(newwin, bd=5)
    E1.place(x=110,y=50)
    L2 = Label(newwin, text="Student Id")
    L2.place(x=10,y=100)
    E2 = Entry(newwin, bd=5)
    E2.place(x=100,y=100)
    sub=Button(newwin,text="Add",command=add_query)
    sub.place(x=120,y=250)

def update_data(): # new window definition
    def UPDD():
        global root
        query = "UPDATE student SET name = %s WHERE id = %s"
        session.execute(query, (E2.get(), E1.get()))
        add.config(state=NORMAL)
        newwin.destroy()

    newwin = Toplevel(root)
    newwin.geometry('400x350')
    newwin.title("Update Student")
```

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

```python
    add.config(state=NORMAL)

    L1 = Label(newwin, text="Student id")
    L1.place(x=10,y=50)
    E1 = Entry(newwin, bd=5)
    E1.place(x=110,y=50)

    L2 = Label(newwin, text="Student Name")
    L2.place(x=10,y=100)
    E2 = Entry(newwin, bd=5)
    E2.place(x=100,y=100)

    sub=Button(newwin,text="Update",command=UPDD)
    sub.place(x=120,y=200)

def del_data():
    def delete():
        global root
        query = "DELETE FROM student WHERE id = %s"
        session.execute(query, (E1.get(),))
        add.config(state=NORMAL)
        newwin.destroy()

    newwin=Toplevel(root)
    newwin.geometry('400x350')
    newwin.title("Delete Student")
    add.config(state=NORMAL)
    L1 = Label(newwin, text="Student Id")
    L1.place(x=10, y=50)
    E1 = Entry(newwin,bd=5)
    E1.place(x=110, y=50)
    sub = Button(newwin, text="Delete Student", command=delete)
    sub.place(x=120, y=200)

def display():
    # Connect to Cassandra cluster
    # cluster = Cluster(['127.0.0.1'])  # Replace with your Cassandra nodes
    # session = cluster.connect('my_keyspace')  # Replace with your keyspace

    # Create a new window
    newwin = Toplevel(root)
    newwin.geometry('400x350')
    newwin.title('Student Details')

    # Add labels for student name and ID
    L1 = Label(newwin, text='Student Name')
    L1.grid(row=0, column=0)
    L2 = Label(newwin, text='Id')
```

NAME:TUSHAR PATIL
PRN:2020BTECS00075 ADS9

```
    L2.grid(row=0, column=1)

    # Retrieve student details from Cassandra database
    rows = session.execute('SELECT * FROM student')
    i = 1
    for row in rows:
        L1 = Label(newwin, text=row.name)
        L1.grid(row=i, column=0)
        L2 = Label(newwin, text=row.id)
        L2.grid(row=i, column=1)
        i += 1

    # Close the Cassandra session and cluster
    # session.shutdown()
    # cluster.shutdown()

# Create buttons for adding, deleting, updating, and showing student details
add = Button(root, text='Add New Student', command=add_course)
delete = Button(root, text='Delete Student', command=del_data)
update = Button(root, text='Update Student Details', command=update_data)
show = Button(root, text='Show Student Details', command=display)
add.place(x=50, y=50)
delete.place(x=50, y=100)
update.place(x=200, y=50)
show.place(x=200, y=100)

root.mainloop()
```