Experiment. 10

Write Python programs to understand GUI database connectivity to perform CRUD operations in python. (Use any one database like SQLite, MySQL, Oracle, PostgreSQL etc.)

Roll No.	01
Name	Aamir Ansari
Class	D10A
Subject	Python Lab
LO Mapped	LO1: Understand the structure, syntax, and semantics of the Python language. LO5: Gain proficiency in evaluating database operations in Python.

Aim: Write Python programs to understand GUI database connectivity to perform CRUD operations in python. (Use any one database like SQLite, MySQL, Oracle, PostgreSQL etc.)

Introduction:

Database: A database is a file that is organized for storing data. Like a dictionary, database software is designed to keep the inserting and accessing of data very fast, even for large amounts of data. Database software maintains its performance by building indexes as data is added to the database to allow the computer to jump quickly to a particular entry. There are many different database systems which are used for a wide variety of purposes including:

1. Oracle
2. MySQL
3. Microsoft SQL Server
4. PostgreSQL
5. SQLite

We focus on MySQL because it is a very common database. MySQL Connector Python is written in pure Python and it is self-sufficient to execute database queries through Python. It is an official Oracle-supported driver to work with MySQL and Python. It is Python 3 compatible and actively maintained. There are following steps to connect a Python application to database:

1. Import mysql.connector module
2. Create the connection object
3. Create cursor object
4. Execute the query

To install the MySQL connector module, use the pip command to install MySQL connector in Python.

pip install mysql-connector-python

To import the MySQL connector module, use the following command so that you can use this module's methods to communicate with the MySQL database.

import mysql.connector

To create a connection between the MySQL database and the python application, the **connect()** method of mysql.connector module is used. We need to know the following details of the MySQL server to perform the connection from Python.

Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

Argument	**Description**
Username	The username that you use to work with MySQL Server. The

	default username for the MySQL database is root.
Password	Password is given by the user at the time of installing the MySQL server. If you are using root then you won't need the password.
Host name	The server name or IP address on which MySQL is running. If you are running on localhost, then you can use localhost or its IP 127.0.0.0
Database name	The name of the database to which you want to connect and perform the operations.

Use the connect() method of the MySQL Connector class with the required arguments to connect MySQL. It would return a MySQLConnection object if the connection established successfully.

Connection-Object = mysql.connector.connect(host = <host-name>, user = <username>, passwd = <password>)

The **connect()** method can throw a Database error exception if one of the required parameters is wrong. For example, if you provide a database name that is not present in MySQL. The **is_connected()** is the method of the MySQLConnection class through which we can verify if our Python application is connected to MySQL.

At last, we are closing the MySQL database connection using a **close()** method of MySQLConnection class.

The cursor object can be defined as an abstraction specified in the Python DB-API 2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

Use the **cursor()** method of a MySQLConnection object to create a cursor object to perform various SQL operations.

<my_cur> = conn.cursor()

The **execute()** methods run the SQL query and return the result. Use **cursor.fetchall()** or **fetchone()** or **fetchmany()** to read query results. Use **cursor.close()** and **connection.close()** method to close open connections after your work completes.

Catch Exception: It may occur during this process by importing the Error class from the MySQL connector module using a from mysql.connector import Error statement. Error class is useful to debug when we fail to connect to MySQL. For example, ACCESS DENIED ERROR when the username or password is wrong.

CRUD Operations:

1. **Insert a single row into MySQL table from Python**: Define a SQL Insert query. Next, prepare a SQL INSERT query to insert a row into a table. In the insert query, we mention column names and their values to insert in a table.

INSERT INTO mysql_table (col1, col2, …) VALUES (val1, val2, …);

Get Cursor Object from Connection by using a **connection.cursor()** method to create a cursor object. This method creates a new MySQLCursor object.

Execute the insert query using the **cursor.execute()** method. This method executes the operation stored in the Insert query.

Commit your changes after the successful execution of a query, changes persist into a database using the **commit()** of a connection class.

Get the number of rows affected after a successful insert operation, use a **cursor.rowcount()** method to get the number of rows affected. The count depends on how many rows you are inserting.

Verify the result using the SQL SELECT query by executing a MySQL select query from Python to see the new changes.

Close the cursor object and database connection object using **cursor.close()** and **connection.close()** method to close open connections after your work completes.

Example:

```
import mysql.connector;
conn=mysql.connector.connect(host='localhost',database='world',user='root',password='st
udent')

if conn.is_connected():
        print('Connected to MySQL database')
cursor=conn.cursor()
str="insert into emptab(eno,ename,sal) values(9999,'srinivas',9999.99)"

try:
        cursor.execute(str)
        conn.commit()
        print('1 row inserted....')
except:
        conn.rollback()
cursor.close()
conn.close()
```

```
Connected to MySQL database
1 row inserted....
```

2. **Select rows from MySQL table from Python**: Define a SQL SELECT Query. Next, prepare a SQL SELECT query to fetch rows from a table. You can select all or limited rows based on your requirement. If the where condition is used, then it decides the number of rows to fetch.

SELECT col1, col2,…colnN FROM MySQL_table WHERE id = 10;

Get Cursor Object from Connection using a **connection.cursor()** method to create a cursor object. This method creates a new MySQLCursor object.

Execute the select query using the **cursor.execute()** method.

Extract all rows from a result after successfully executing a Select operation, use the **fetchall()** method of a cursor object to get all rows from a query result. It returns a list of rows.

Iterate a row list using a for loop and access each row individually (Access each row's column data using a column name or index number).

Close the cursor object and database connection object using **cursor.close()** and **connection.close()** method to close open connections after your work completes.

Example:

```
import mysql.connector;
conn=mysql.connector.connect(host='localhost',database='world',user='root',password='st
udent')

if conn.is_connected():
        print('Connected to MySQL database')
cursor=conn.cursor()
cursor.execute("select * from emptab")

rows=cursor.fetchall()
print('Total number of rows=',cursor.rowcount)
for row in rows:
        eno=row[0]
        ename=row[1]
        sal=row[2]
        print('%-6d %-15s %10.2f'% (eno,ename,sal))
cursor.close()
conn.close()
```

```
Connected to MySQL database
Total number of rows= 2
1001   Nagesh          7800.00
1002   gagesh          8800.00
```

3. **Update a row from MySQL table from Python**: Prepare a SQL Update Query. Prepare an update statement query with data to update.

UPDATE table_name SET col1 = val1, col2 = val2 WHERE condition;

Execute the UPDATE query using **cursor.execute()** method. This method executes the operation stored in the UPDATE query.

Commit your changes and make modification persistent into a database using the **commit()** of a connection class.

Extract the number of rows affected after a successful update operation, use a **cursor.rowcount()** method to get the number of rows affected. The count depends on how many rows you are updating.

Verify the result using the SQL SELECT query by executing a MySQL select query from Python to see the new changes.

Close the cursor object and database connection object using **cursor.close()** and **connection.close()** method to close open connections after your work completes.

Example:

```
import mysql.connector;

conn=mysql.connector.connect(host='localhost',database='world',user='root',password='st
udent')

cursor=conn.cursor()

def update_rows(eno):
        str="update emptab set sal=sal+1000 where
        eno='%d'"
        args=(eno)

        try:
                cursor.execute(str%args)
                conn.commit()
                print('1 row updated....')
```

```
        except:
            conn.rollback()
        finally:
            cursor.close()
            conn.close()

x=int(input('Enter eno:'))
update_rows(x)
```

```
Enter eno:1002
1 row updated....
```

4. **Delete a single row from MySQL table from Python**: Define a SQL Delete Query. Next, prepare a SQL delete query to delete a row from a table. Delete query contains the row to be deleted based on a condition placed in where clause of a query.

DELETE FROM MySQL_table WHERE id=10;

Get Cursor Object from Connection using a **connection.cursor()** method to create a cursor object. This method creates a new MySQLCursor object.

Execute the delete query using the **cursor.execute()** method. This method executes the operation stored in the delete query. After a successful delete operation, the **execute()** method returns us the number of rows affected.

Commit your changes after successfully executing a delete operation, make changes persistent into a database using the **commit()** of a connection class.

Get the number of rows affected by using a **cursor.rowcount()** method to get the number of rows affected. The count depends on how many rows you are deleting. You can also execute a MySQL select query from Python to verify the result.

Close the cursor object and database connection object using **cursor.close()** and **connection.close()** method to close open connections after your work completes.

Example:

```
import mysql.connector;

conn=mysql.connector.connect(host='localhost',database='world',user='root',password='st
udent')
cursor=conn.cursor()

def delete_rows(eno):
    str="delete from emptab where eno='%d'"
    args=(eno)
```

```
        try:
            cursor.execute(str%args)
            conn.commit()
            print('1 row deleted....')
        except:
            conn.rollback()
        finally:
            cursor.close()
            conn.close()

x=int(input('Enter eno:'))
delete_rows(x)
```

```
Enter eno:1001
1 row deleted....
```

Results:

Code:

```
from tkinter.ttk import *
from tkinter import *
import mysql.connector
from tkinter import messagebox

mydb=mysql.connector.connect(
   host="localhost",
   user="root",
   password="",
   database="python_class"
)

mycursor=mydb.cursor()
root=Tk()
root.title("Student Details")
root.geometry("850x400")
root.configure(background="#f3efd6")
root.configure(highlightbackground="#d9d9d9")
root.configure(highlightcolor="black")

label1 = Label(root, text="Roll No.", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=1, column=0, padx=(10,0))
label2 = Label(root, text="First Name", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=1, column=4, padx=(10,0))
label3 = Label(root, text="Last Name", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=2, column=0, padx=(10,0))
```

```
label4 = Label(root, text="Phone Number", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=2, column=4, padx=(10,0))
label5 = Label(root, text="city", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=3, column=0, padx=(10,0))
label6 = Label(root, text="state", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=3, column=4, padx=(10,0))
label7 = Label(root, text="age", width=13, anchor='w', font=("Sylfaen", 12)).grid(row=4, column=2, padx=(10,0))

e1 = Entry(root, width=20)
e1.grid(row=1, column=1, padx=(0,10), pady = 20)
e2 = Entry(root, width=20)
e2.grid(row=1, column=5, padx=(0,10), pady = 20)
e3 = Entry(root, width=20)
e3.grid(row=2, column=1, padx=(0,10), pady = 20)
e4 = Entry(root, width=20)
e4.grid(row=2, column=5, padx=(0,10), pady = 20)
e5 = Entry(root, width=20)
e5.grid(row=3, column=1, padx=(0,10), pady = 20)
e6 = Entry(root, width=20)
e6.grid(row=3, column=5, padx=(0,10), pady = 20)
e7 = Entry(root, width=20)
e7.grid(row=4, column=3, padx=(0,10), pady = 20)

def Register():
    roll_no=e1.get()
    dbroll_no=""
    Select="select roll_no from student where roll_no='%s'" %(roll_no)
    mycursor.execute(Select)
    result=mycursor.fetchall()
    for i in result:
        dbroll_no=i[0]
    if(roll_no == dbroll_no):
        messagebox.askokcancel("Information","Record Already exists")
    else:
                Insert="Insert into student(roll_no,f_name,l_name,phone_number,city,state,age) values(%s,%s,%s,%s,%s,%s,%s)"
        f_name=e2.get()
        l_name=e3.get()
        phone_number=e4.get()
        city=e5.get()
        state=e6.get()
        age=e7.get()
         if(f_name !="" and l_name !="" and phone_number !="" and city !="" and state !="" and age !=""):
           Value=(roll_no,f_name,l_name,phone_number,city,state,age)
```

```
            mycursor.execute(Insert,Value)
            mydb.commit()
            messagebox.askokcancel("Information","Record inserted")
            e1.delete(0, END)
            e2.delete(0, END)
            e3.delete(0, END)
            e4.delete(0, END)
            e5.delete(0, END)
            e6.delete(0, END)
            e7.delete(0, END)
        else:
             if (f_name == "" and l_name == "" and phone_number == "" and city == "" and state ==
"" and age == ""):
             messagebox.askokcancel("Information","New Entry Fill All Details")
            else:
             messagebox.askokcancel("Information", "Some fields left blank")

def ShowRecord():
    roll_no=e1.get()
    print(roll_no)
    dbroll_no=""
    Select="select roll_no from student where roll_no='%s'" %(roll_no)
    mycursor.execute(Select)
    result1=mycursor.fetchall()
    for i in result1:
        dbroll_no=i[0]
    Select1="select f_name,l_name,phone_number,city,state,age from student where roll_no='%s'"
%(roll_no)
    mycursor.execute(Select1)
    result2=mycursor.fetchall()
    f_name=""
    l_name=""
    phone_number=""
    city=""
    state=""
    age=""
    if(str(roll_no) == str(dbroll_no)):
        for i in result2:
            f_name=i[0]
            l_name=i[1]
            phone_number=i[2]
            city=i[3]
            state=i[4]
            age=i[5]
        e2.insert(0,f_name)
        e3.insert(0, l_name)
```

```
        e4.insert(0, phone_number)
        e5.insert(0, city)
        e6.insert(0, state)
        e7.insert(0, age)
    else:
        messagebox.askokcancel("Information","No Record exists")

def Delete():
    roll_no=e1.get()
    Delete="delete from student where roll_no='%s'" %(roll_no)
    mycursor.execute(Delete)
    mydb.commit()
    messagebox.showinfo("Information","Record Deleted")
    e1.delete(0,END)
    e2.delete(0, END)
    e3.delete(0, END)
    e4.delete(0, END)
    e5.delete(0, END)
    e6.delete(0, END)
    e7.delete(0,END)

def Update():
    roll_no=e1.get()
    f_name=e2.get()
    l_name=e3.get()
    phone_number=e4.get()
    city=e5.get()
    state=e6.get()
    age=e7.get()
        Update="Update student set f_name='%s', l_name='%s', phone_number='%s', city='%s',
state='%s', age='%s' where roll_no='%s'"
%(f_name,l_name,phone_number,city,state,age,roll_no)
    mycursor.execute(Update)
    mydb.commit()
    messagebox.showinfo("Info","Record Update")

def Showall():
    class A(Frame):
        def __init__(self, parent):
            Frame.__init__(self, parent)
            self.CreateUI()
            self.LoadTable()
            self.grid(sticky=(N, S, W, E))
            parent.grid_rowconfigure(0, weight=1)
            parent.grid_columnconfigure(0, weight=1)
```

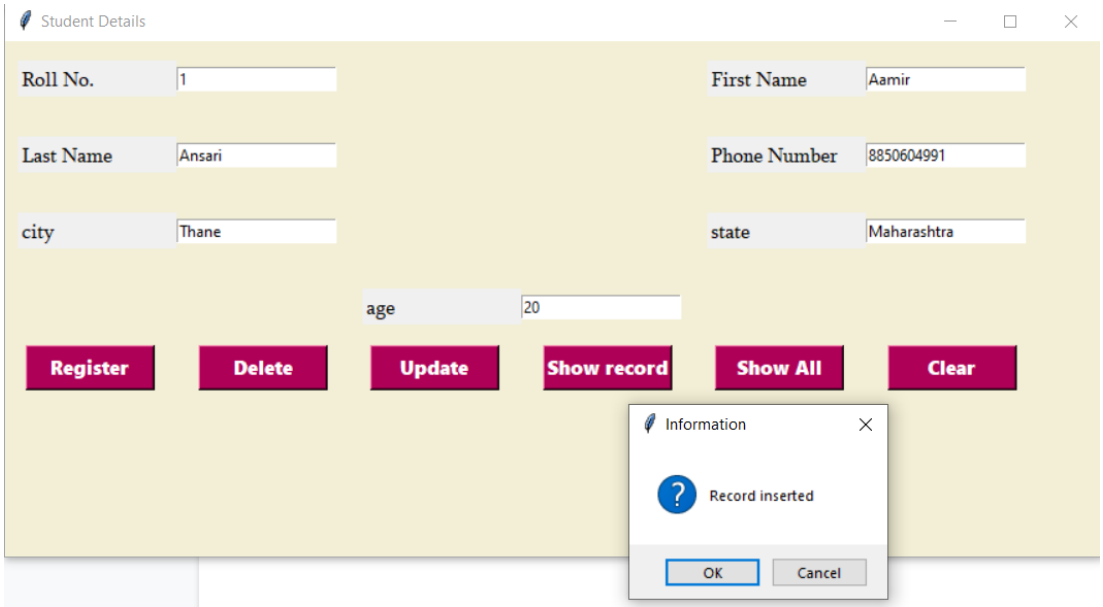
```
    def CreateUI(self):
        tv=Treeview(self)
        tv['columns']=('1', '2', '3', '4', '5', '6', '7')
        tv['show']='headings'

        tv.column("1", width = 90, anchor ='c')
        tv.column("2", width = 90, anchor ='c')
        tv.column("3", width = 90, anchor ='c')
        tv.column("4", width = 90, anchor ='c')
        tv.column("5", width = 90, anchor ='c')
        tv.column("6", width = 90, anchor ='c')
        tv.column("7", width = 90, anchor ='c')

        tv.heading('1', text='Roll No.')
        tv.heading('2', text='First Name')
        tv.heading('3', text='Last Name')
        tv.heading('4', text='Phone Number')
        tv.heading('5', text='city')
        tv.heading('6', text='state')
        tv.heading('7', text='age')

        tv.grid(sticky=(N,S,W,E))
        self.tview = tv
        self.grid_rowconfigure(0, weight=1)
        self.grid_columnconfigure(0, weight=1)

    def LoadTable(self):
        Select="Select * from student"
        mycursor.execute(Select)
        result=mycursor.fetchall()
        roll_no=""
        f_name=""
        l_name=""
        phone_number=""
        city=""
        state=""
        age=""
        for i in result:
            roll_no=i[0]
            f_name=i[1]
            l_name=i[2]
            phone_number=i[3]
            city=i[4]
            state=i[5]
            age=i[6]
```

```
self.tview.insert("",'end',text="L1",values=(roll_no,f_name,l_name,phone_number,city,state,age)
)
    root=Tk()
    root.title("Overview Page")
    root.resizable(width = 1, height = 1)
    A(root)

def Clear():
    e1.delete(0, END)
    e2.delete(0, END)
    e3.delete(0, END)
    e4.delete(0, END)
    e5.delete(0, END)
    e6.delete(0, END)
    e7.delete(0, END)

button1 = Button(root, text="Register", width=10, bg="#06a099",
command=Register,activebackground="#9d004f", activeforeground="white",
background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size
11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9",
highlightcolor="black")
button1.grid(row=8,column=0)
button2 = Button(root, text="Delete", width=10, bg="#06a099",
command=Delete,activebackground="#9d004f", activeforeground="white",
background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size
11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9",
highlightcolor="black")
button2.grid(row=8,column=1)
button3 = Button(root, text="Update", width=10, bg="#06a099",
command=Update,activebackground="#9d004f", activeforeground="white",
background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size
11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9",
highlightcolor="black")
button3.grid(row=8,column=2)
button4 = Button(root, text="Show record", width=10, bg="#06a099",
command=ShowRecord,activebackground="#9d004f", activeforeground="white",
background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size
11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9",
highlightcolor="black")
button4.grid(row=8,column=3)
button5 = Button(root, text="Show All", width=10, bg="#06a099",
command=Showall,activebackground="#9d004f", activeforeground="white",
background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size
11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9",
highlightcolor="black")
```

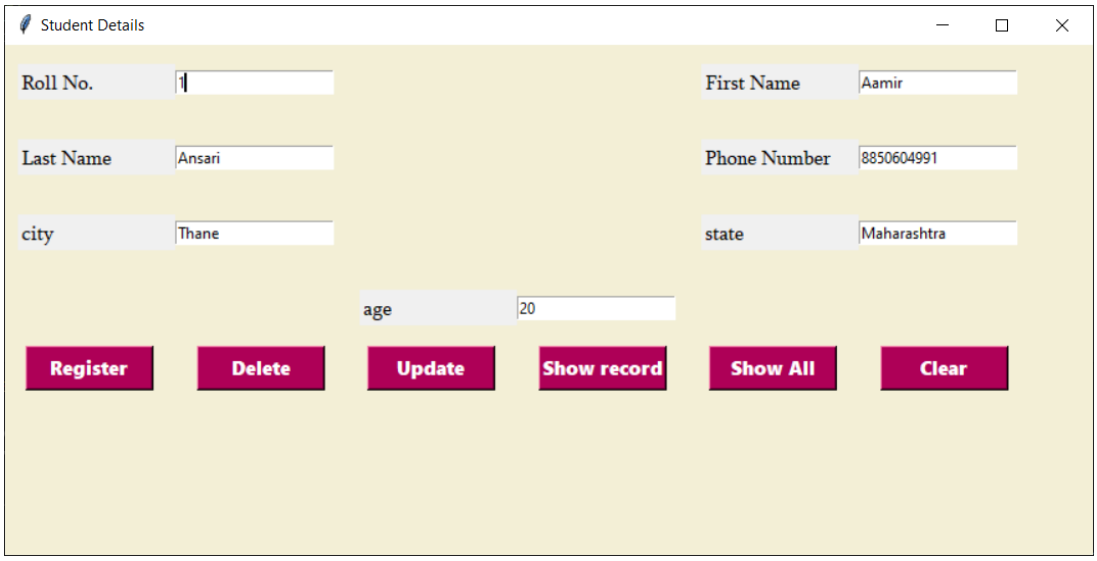
```
button5.grid(row=8,column=4)
button6 = Button(root, text="Clear", width=10, bg="#06a099", command=Clear,activebackground="#9d004f", activeforeground="white", background="#ae0057", disabledforeground="#a3a3a3", font="-family {Segoe UI Black} -size 11 -weight bold", foreground="#ffffff", highlightbackground="#d9d9d9", highlightcolor="black")
button6.grid(row=8,column=5)

root.mainloop()
```

Output:

Create

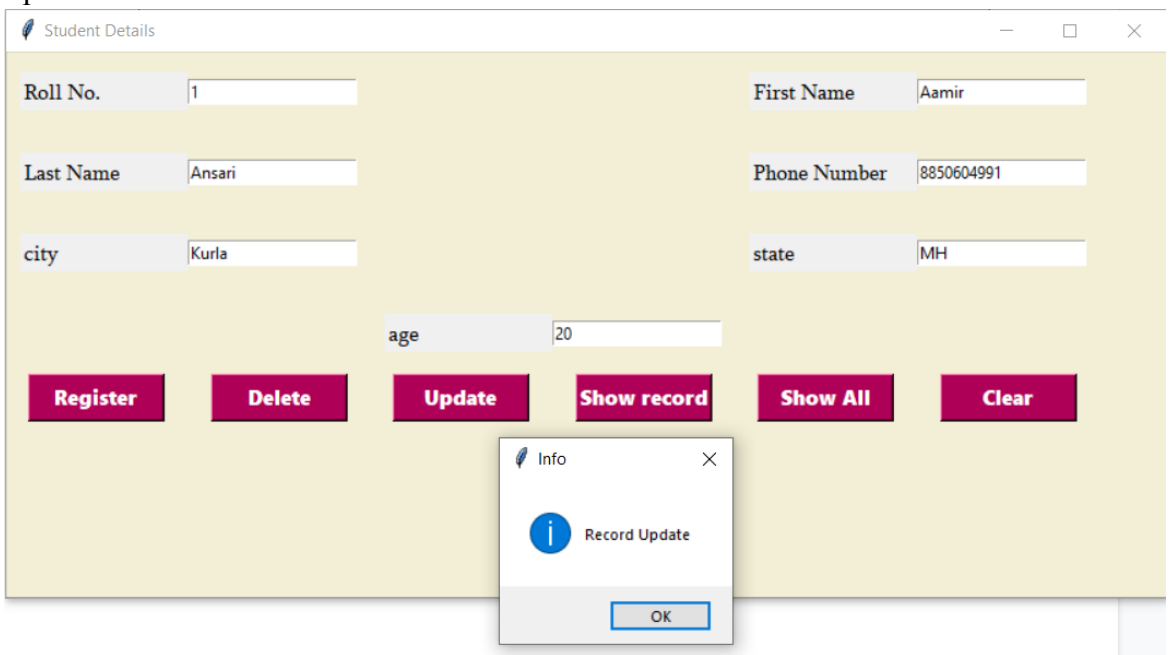


Show record

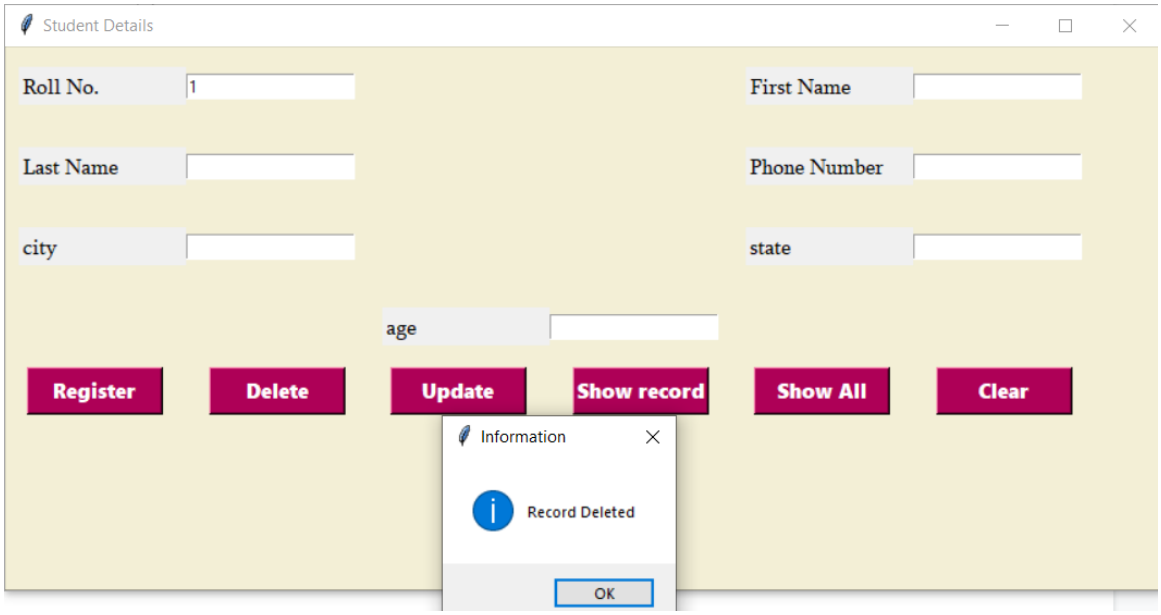


Show all records

Update



Delete



Conclusion:
Hence we have successfully understood the concepts of database connectivity in Python with its GUI, by performing hands on operation of Create, Read, Update and Delete