MyPyGui

Python and

MySql

Interface

Name: ***Ujjwal Kakar***

**CERTIFICATE**

CLASS: XII-F YEAR: 2022-2023

This is to certify that Investigatory Project is successfully completed by Ujjwal Kakar of Class: XII F for the academic year 2022-2023.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Subject Teacher’s Signature Externam Examiner Signature

**Acknowledgement:**

I, Ujjwal Kakar of class XII- F would like to express our sincere gratitude to my computer science teacher Ms. Gurmeet Kaur, PGT Computer Science, for her **constant** support and ideas for growth in my endeavour to pursue thoughtful and functional programs as we study python.

I also made use well of online tools to help guide us with more complex parts of syntax and documentation in labriaries, such as time.

**INDEX**

|  |  |
| --- | --- |
| 1 | *Brief Overview of Project* |
| 2 | ***Limitations of Project*** |
| 3 | ***Source Code of Project*** |
| 4 | ***Output Screens*** |
| 4 | ***Future Enhancement of Project*** |
| 5 | ***Bibliography*** |

**Brief Overview of Project:**

**Definition:**

This Project Aims to be able to create an easy to use and understandable graphical user interface as a frontend through python, for a simple user to be able to use MySQL

**Objective:**

Connecting MySQL with Python to create, modify, delete and view, databases, tables, and records in conjunction with a visually pleasing GUI

**Making Detail:**

The project “MyPyGui” is developed by Ujjwal Kakar, it took approx. 2 days to develop this project, working 4 hours a day. All functions and datasets completed by us only as per my view and knowledge.

**Reason for choosing the Topic:**

Databases are a great asset to be able to store and access data. MySQL is a very popular database system, but it is very syntax based and difficult for a non-technical person.

For the lay man, such technologies are still difficult to use and access.

Hence this project is created to be able to provide an easy to use intuitive frontend, as well as a working and capable backend, to the normal person. This will increase the reach of technology, provide growth and allow for more efficient data storage

All in all, it is capable to improve lives

**Hardware Requirements:**

A Computer/Laptop with

Operating System-Windows 7 or above

5 GB free disk space.

**Software Requirements:**

Python 3.9

sd library (Visual Studio Correction Library)

time library

PySimpleGUI library

Os Library

**Limitations of project:**

* It is not web based project
* No provision to print hard copies of student data
* It cannot access the core SQL serrvices like foreign key.
* It is limited when against more complex queries and syntaxes.

**Source Code of Project:**

**funcDec.py**

import mysql.connector as mq

import PySimpleGUI as sg

def checkCredentials(h,u,p):

try:

conn = mq.connect(host=h,user=u,password=p)

if conn.is\_connected():

print("Successful Connection")

return conn

else:

print("Connection Failed, wrong details")

return False

except:

print("Connection Failed, wrong details")

return False

def getdatabases(conn):

c= conn.cursor()

c.execute("SHOW DATABASES")

return [item[0] for item in c]

def loaddb(conn,db):

c= conn.cursor()

c.execute("USE "+db)

c.execute("SHOW TABLES")

return [item[0] for item in c]

def createdb(conn,db):

c= conn.cursor()

c.execute("CREATE DATABASE "+db)

conn.commit()

def removedb(conn,db):

c= conn.cursor()

c.execute("DROP DATABASE "+db)

conn.commit()

def createtable(conn,table,param):

c= conn.cursor()

c.execute("CREATE TABLE "+table+"("+param+")")

conn.commit()

def removetable(conn,table):

c= conn.cursor()

c.execute("DROP TABLE "+table)

conn.commit()

def tablecols(conn, table):

c=conn.cursor()

c.execute("DESC "+table)

return [item[0] for item in c]

def inserttable(conn,table,data):

c=conn.cursor()

try:

query= "INSERT INTO "+table+" VALUES("+",".join(data)+")"

c.execute(query)

except: sg.popup("Data not added",size=(120,90),modal=True,auto\_close=True,auto\_close\_duration=3)

conn.commit()

def findprimarykey(conn,table):

c=conn.cursor()

c.execute("DESC "+table)

recvdat = [x for x in c]

print(recvdat)

for x in range(len(recvdat)):

if recvdat[x][3]=="PRI":

return (recvdat[x][0],x)

else:

return (recvdat[0][0],0)

def deleteintable(conn,table,data):

c=conn.cursor()

pridata=findprimarykey(conn,table)

print(pridata)

try: c.execute("Delete from "+table+" where "+pridata[0]+"="+data[pridata[1]])

except: c.execute("Delete from "+table+" where "+pridata[0]+"=\""+data[pridata[1]]+"\"")

conn.commit()

def selectfromtable(conn,table):

c=conn.cursor()

c.execute("SELECT \* FROM "+table)

return [[str(y) for y in x] for x in c]

def tableformatting(conn,dat,table):

dat = [tablecols(conn,table)]+dat

maxl = max([len(line) for line in dat])

for line in dat:

print(line)

while len(line)!=maxl:

line.append('')

# print(dat)

maxlist = []

for times in range(len(line)):

maxval = 0

for line in dat:

if len(line[times]) > maxval:

maxval = len(line[times])

maxlist.append(maxval)

strR = "|"

for length in maxlist:

strR = strR + r"%-"+str(length)+"s|"

# print("\_"\*(sum(maxlist)+len(maxlist)+1))

out ="‾"\*(sum(maxlist)+len(maxlist))+'\n'

out += str(strR)%tuple(dat[0]) + "\n"

out += "\_"\*(sum(maxlist)+len(maxlist)+1) + "\n"

for line in dat[1:]:

# print(line)

# print(str(strR)%tuple(line))

out+= str(strR)%tuple(line) + "\n"

# print("‾"\*(sum(maxlist)+len(maxlist)))

out +="‾"\*(sum(maxlist)+len(maxlist))+'\n'

return out

**main.py**

from funcDec import \*

import PySimpleGUI as sg,os

from time import sleep

icon = os.getcwd() + r"\classXII\project\prj2\download.ico"

def runwindow1():

sg.theme("Light Blue 3")

layout1 =[[sg.Text("MySQL Connectivity")],

[sg.Text("Connection Status:"),sg.Text("\"Not Connected\"",key="connstatus")],

[sg.Text("Host: "),sg.Input(key="host",font=("Calibri",14))],

[sg.Text("User: "),sg.Input(key="user",font=("Calibri",14))],

[sg.Text("Pass: "),sg.Input(key="pass",font=("Calibri",14))],

[sg.Button("Login"),sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,480),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event == "Login":

global conn

conn = checkCredentials(values["host"],values["user"],values["pass"])

if conn == False: sg.popup\_error("No Connection Established, Wrong Crendentials, Try Again")

else:

window["connstatus"].update("\"Connected\"")

sleep(1)

window.close()

runwindow2()

window.close()

def runwindow2():

sg.theme("Light Blue 3")

dblist=getdatabases(conn)

db\_buttons = [[sg.Button(x,size=(20,1),font=("Calibri",14))] for x in dblist]

layout2 =[[sg.Text("MySQL Status: Connected"),sg.Button("Refresh"),sg.Text(" "\*65),sg.Button("Back")],

[sg.Text("Create Database: "), sg.Button("New Database"),sg.Text(" "\*8),sg.Text("Delete Database: "), sg.Button("Remove")],

[],

[sg.Text("Choose from Existing Database")],]+ db\_buttons + [[sg.Exit()]]

window = sg.Window("SQL Project",layout2,size=(1080,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event in dblist:

window.close()

runwindow3(event)

if event == "Refresh":

window.close()

runwindow2()

if event == "New Database":

runcreatedbwindow()

window.close()

runwindow2()

if event == "Remove":

runremovedbwindow(dblist)

window.close()

runwindow2()

if event == "Back":

window.close()

runwindow1()

window.close()

def runwindow3(db):

dbname=db

sg.theme("Light Blue 3")

tablelist = loaddb(conn,db)

tablebuttons = [[sg.Button(x,size=(20,2),font=("Calibri",14))] for x in tablelist]

layout3 =[[sg.Text("MySQL Status: Connected"),sg.Button("Refresh"),sg.Text(" "\*65),sg.Button("Back")],

[sg.Text("Database Selected: "+dbname)],

[sg.Text("Table Selections")],

[sg.Text("Create Table: "), sg.Button("New Table"),sg.Text(" "\*8),sg.Text("Delete Table: "), sg.Button("Remove")],

[sg.Text(" ")],

[sg.Text("Choose from Existing Table")],

] + tablebuttons + [[sg.Text(" ")],[sg.Text("\_"\*50)],[sg.Exit()]]

window = sg.Window("SQL Project",layout3,size=(1080,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event in tablelist:

window.close()

runwindow4(dbname,event)

if event == "Refresh":

window.close()

runwindow3(dbname)

if event == "New Table":

runcreatetablewindow()

window.close()

runwindow3()

if event == "Remove":

runremovetablewindow(tablelist)

window.close()

runwindow3(dbname)

if event == "Back":

window.close()

runwindow2()

window.close()

def runwindow4(db,table):

sg.theme("Light Blue 3")

layout3 =[[sg.Text("MySQL Status: Connected"),sg.Button("Refresh"),sg.Text(" "\*65),sg.Button("Back")],

[sg.Text("Table: "),sg.Text(table)],

[sg.Text("Table Entry: "), sg.Button("Enter Here"),sg.Text(" "\*8),sg.Text("Delete Entry: "), sg.Button("Delete Here")],

[sg.Text("Full List: "), sg.Button("Click Here")],

[sg.Text(" ")],

[sg.Text("\_"\*50)],

[sg.Exit()]]

window = sg.Window("SQL Project",layout3,size=(1080,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event == "Refresh":

window.close()

runwindow4(table)

if event == "Enter Here":

entertablewindow(db,table)

if event == "Delete Here":

deletetablewindow(db,table)

if event == "Click Here":

listtablewindow(db,table)

if event == "Back":

window.close()

runwindow3(db)

window.close()

def runcreatedbwindow():

sg.theme("Light Blue 3")

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Create Database: ")],

[sg.Text("Database Name: "),sg.Input(key="dbname",font=("Calibri",14))],

[sg.Button("Create"),sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,480),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event == "Create":

createdb(conn,values['dbname'])

window.close()

def runremovedbwindow(dblist):

sg.theme("Light Blue 3")

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Remove Database: ")],]+[[sg.Button(x,size=(20,1),font=("Calibri",14))] for x in dblist]+[[sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event in dblist:

removedb(conn,event)

window.close()

def runcreatetablewindow():

sg.theme("Light Blue 3")

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Create Table: ")],

[sg.Text("Table Name: "),sg.Input(key="tablename",font=("Calibri",14),size=(18,1))],

[sg.Text("Parameters: "),sg.Input(key="tableparam",font=("Calibri",14),size=(60,4))],

[sg.Button("Create"),sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,480),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event == "Create":

createtable(conn,values['tablename'],values['tableparam'])

window.close()

def runremovetablewindow(tablelist):

sg.theme("Light Blue 3")

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Remove table: ")],]+[[sg.Button(x,size=(20,1),font=("Calibri",14))] for x in tablelist]+[[sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event in tablelist:

removetable(conn,event)

window.close()

def entertablewindow(table):

sg.theme("Light Blue 3")

entries=[[sg.Text(item+":"),sg.Input(key=item,font=("Calibri",14))] for item in tablecols(conn,table)]

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Enter into table: '"+table+"' (use double quotes for string)")]]+entries+[[sg.Button("Input"),sg.Exit()],]

window = sg.Window("SQL Project",layout1,size=(720,480),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event == "Input":

inserttable(conn,table,[values[k] for k in values])

window.close()

window.close()

def listtablewindow(db, table):

sg.theme("Light Blue 3")

data = tableformatting(conn,selectfromtable(conn,table),table).split("\n")

formattedtable = [[sg.Text(x,font=("Cascadia Code",14))] for x in data]

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Viewing table: '"+table+"'")],[sg.Text(" ")]]+formattedtable+[[sg.Exit()]]

window = sg.Window("SQL Project",layout1,size=(1280,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

window.close()

def deletetablewindow(db, table):

sg.theme("Light Blue 3")

data = tableformatting(conn,selectfromtable(conn,table),table).split("\n")

formattedtable = [[sg.Text(x,font=("Cascadia Code",14))] for x in data[0:3]]+[

[sg.Text(x,font=("Cascadia Code",14)),sg.Button("Remove",key=x)] for x in data[3:-2]

]+[[sg.Text(x,font=("Cascadia Code",14))] for x in data[-2:-1]]

layout1 =[[sg.Text("MySQL Status: Connected")],

[sg.Text("Viewing table: '"+table+"'")],[sg.Text(" ")]]+formattedtable+[[sg.Exit()]]

window = sg.Window("SQL Project",layout1,size=(1280,720),font=("Eras Bold ITC",20),resizable=True,icon=icon)

while True:

event , values = window.Read()

print(event,values)

if event in (sg.WINDOW\_CLOSED,"Exit"): break

if event in data:

actualEntry=[x.rstrip() for x in event.split("|")][1:-1]

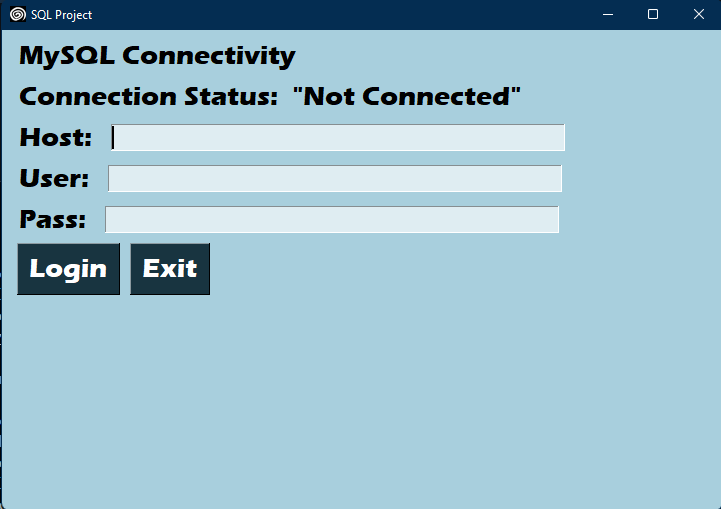
deleteintable(conn,table,actualEntry)

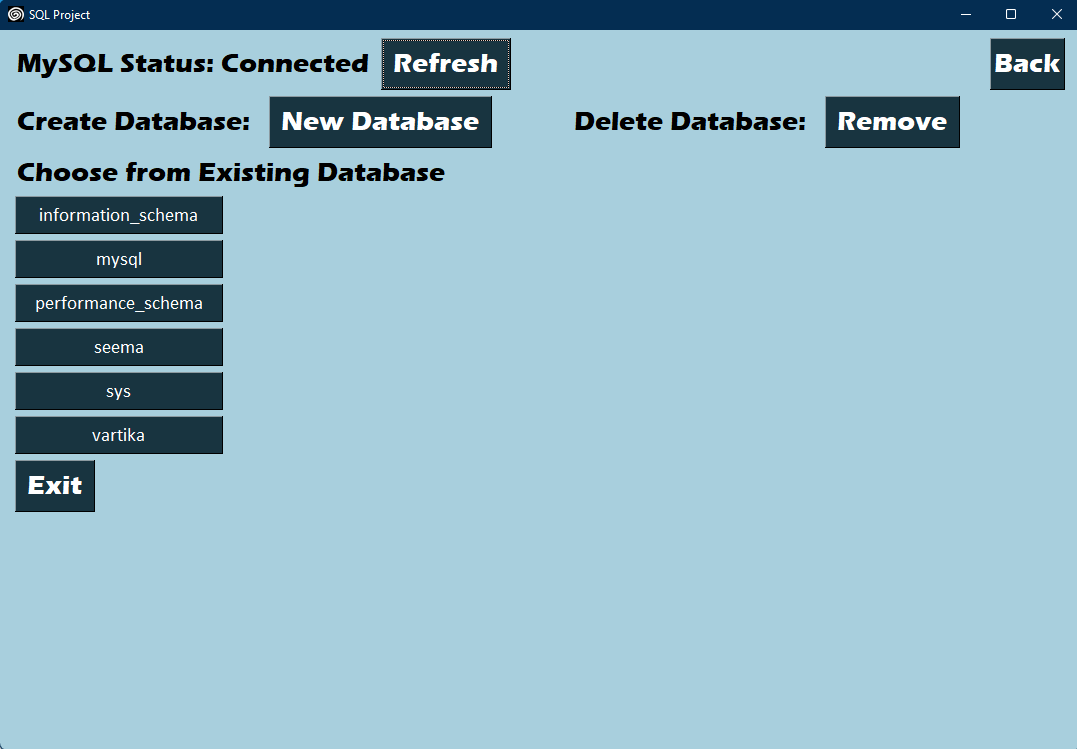
window.close()

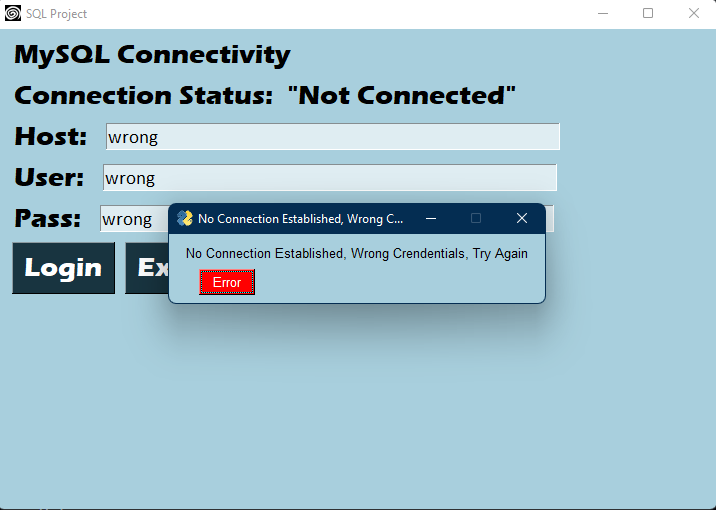
window.close()

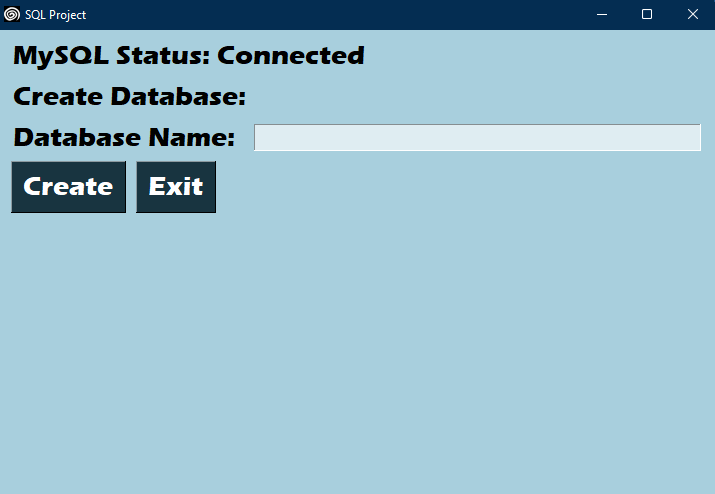
runwindow1()

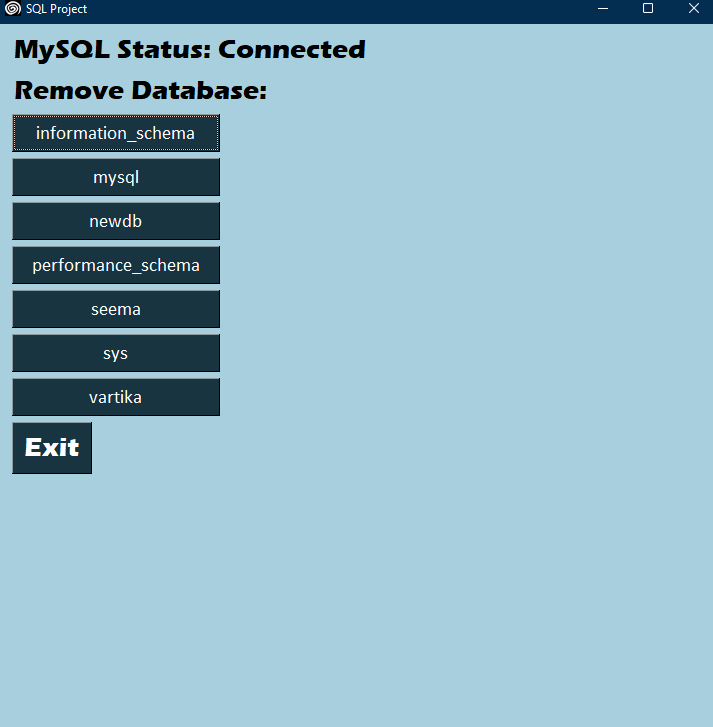
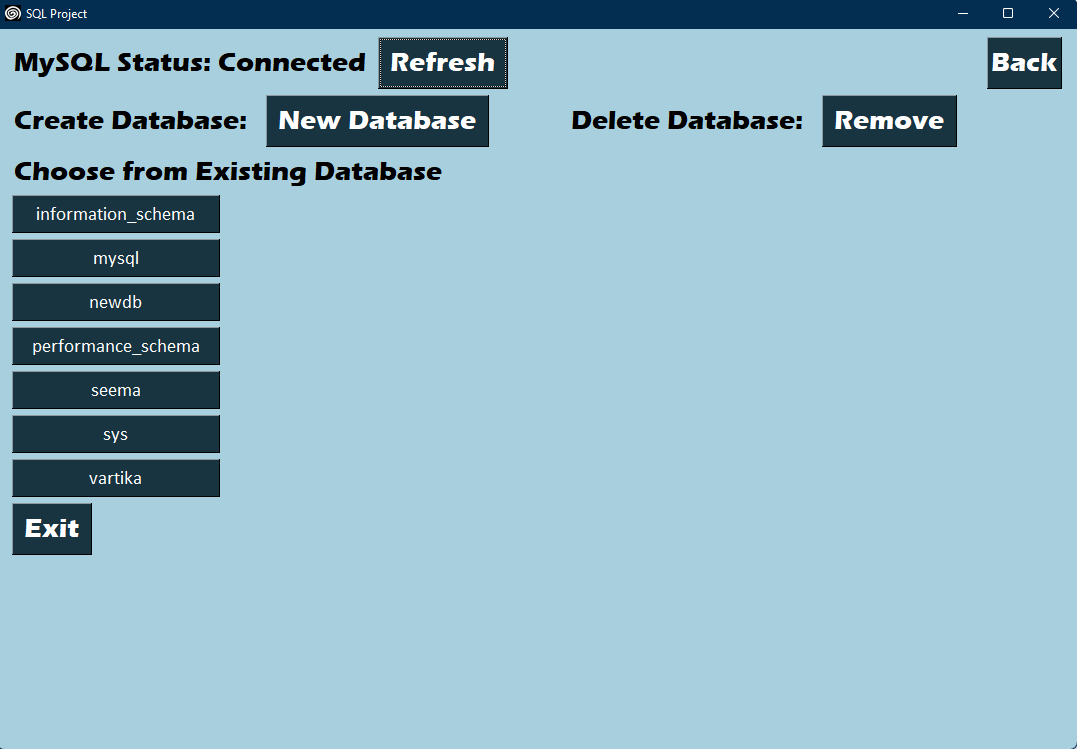
**Output Screens:**

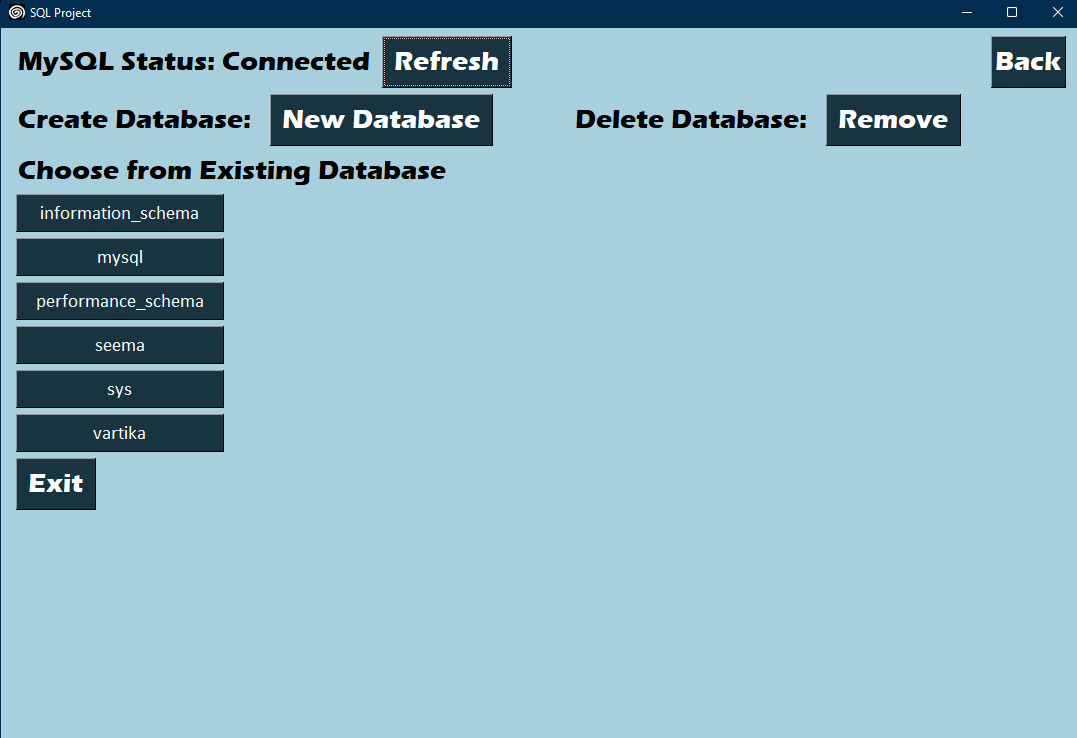
****

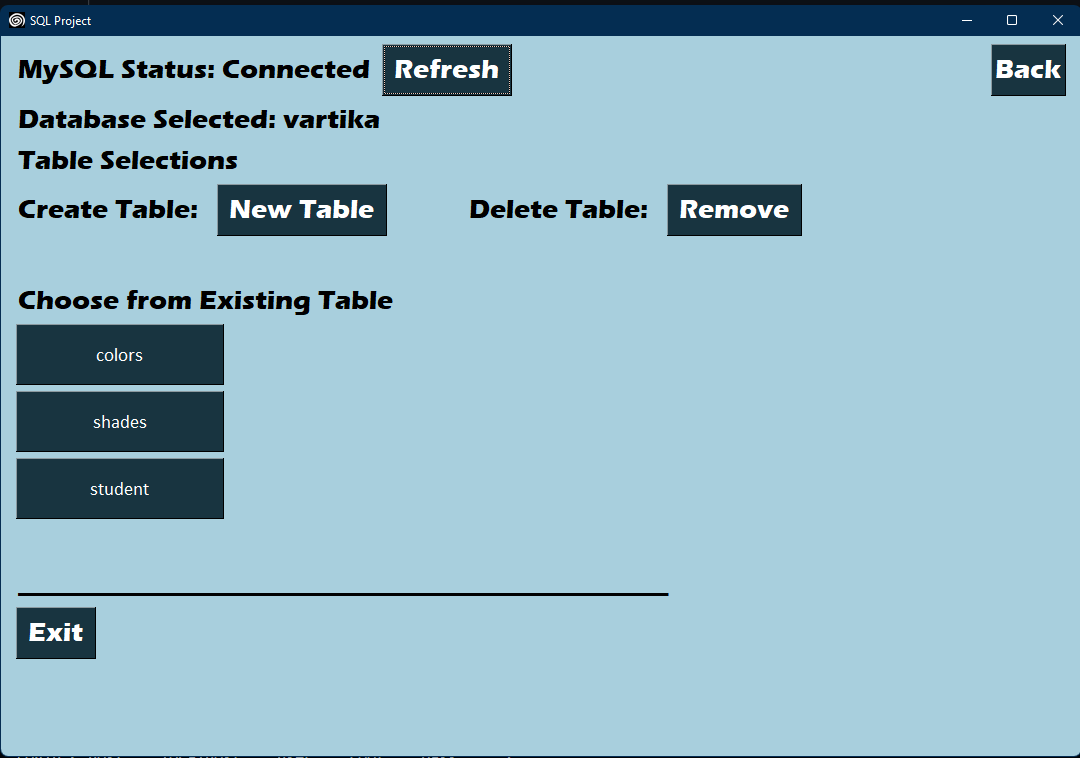


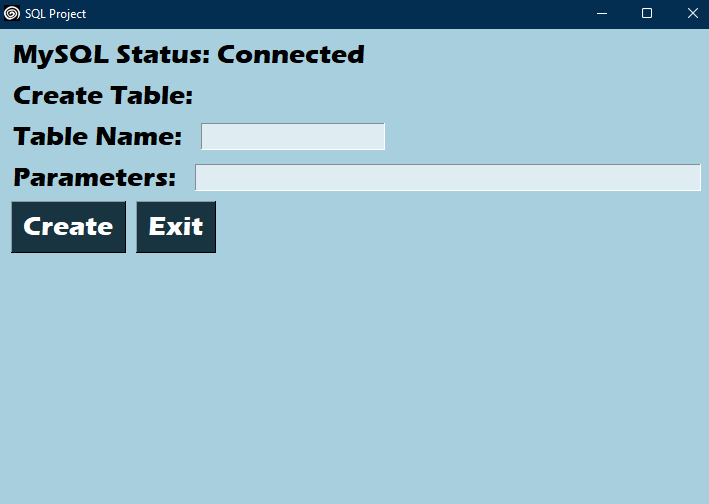


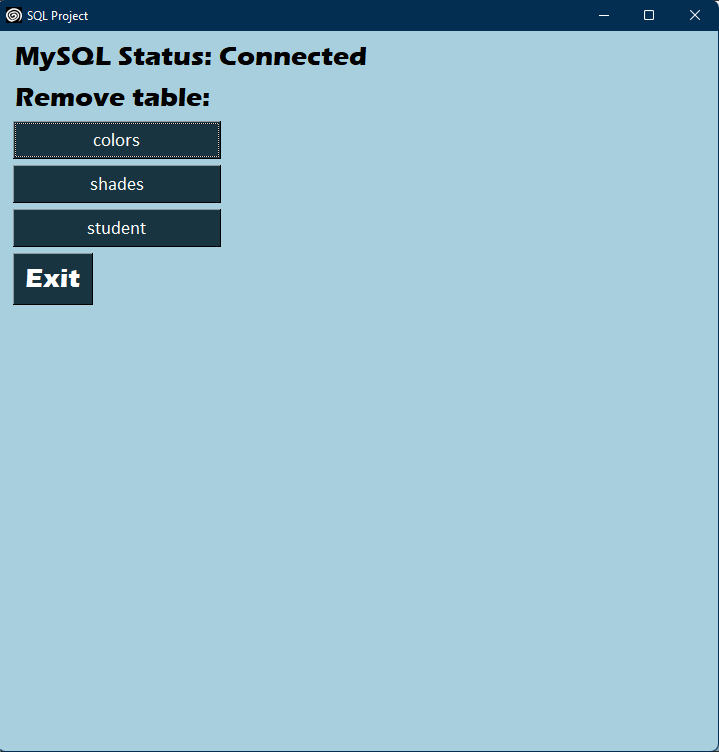


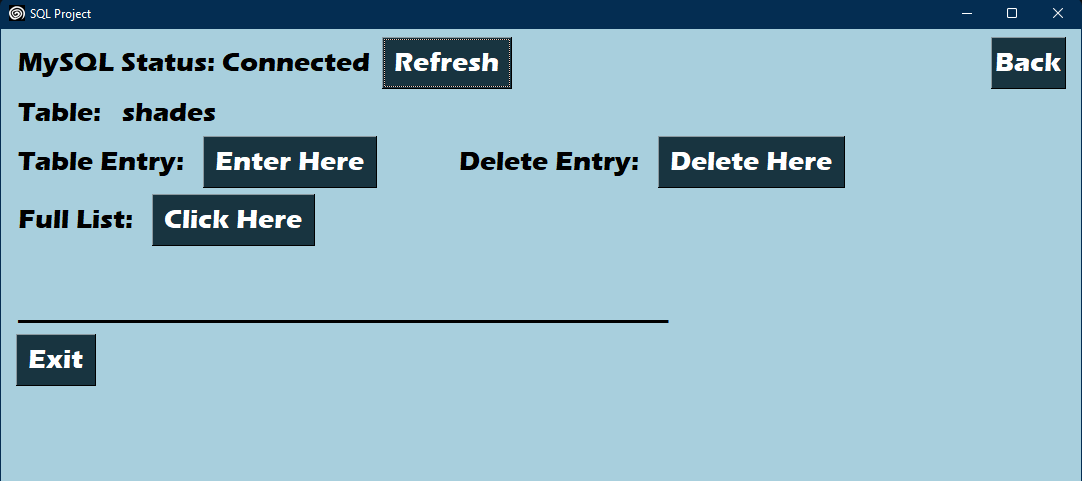


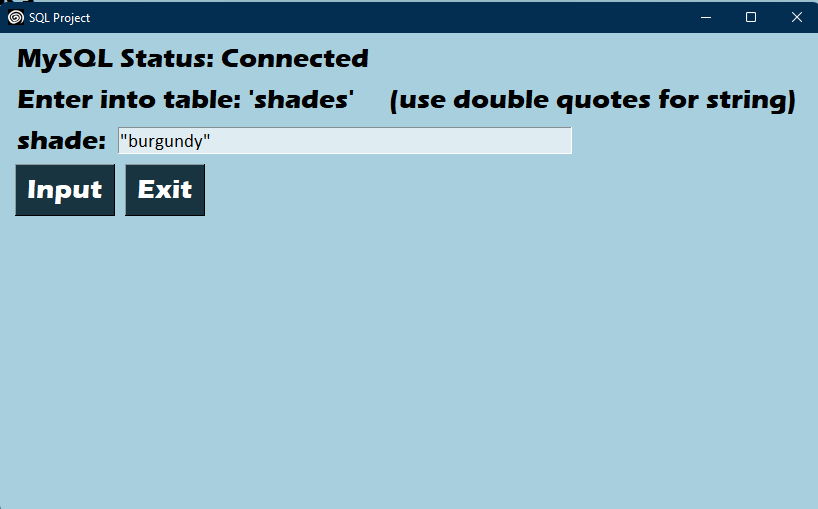


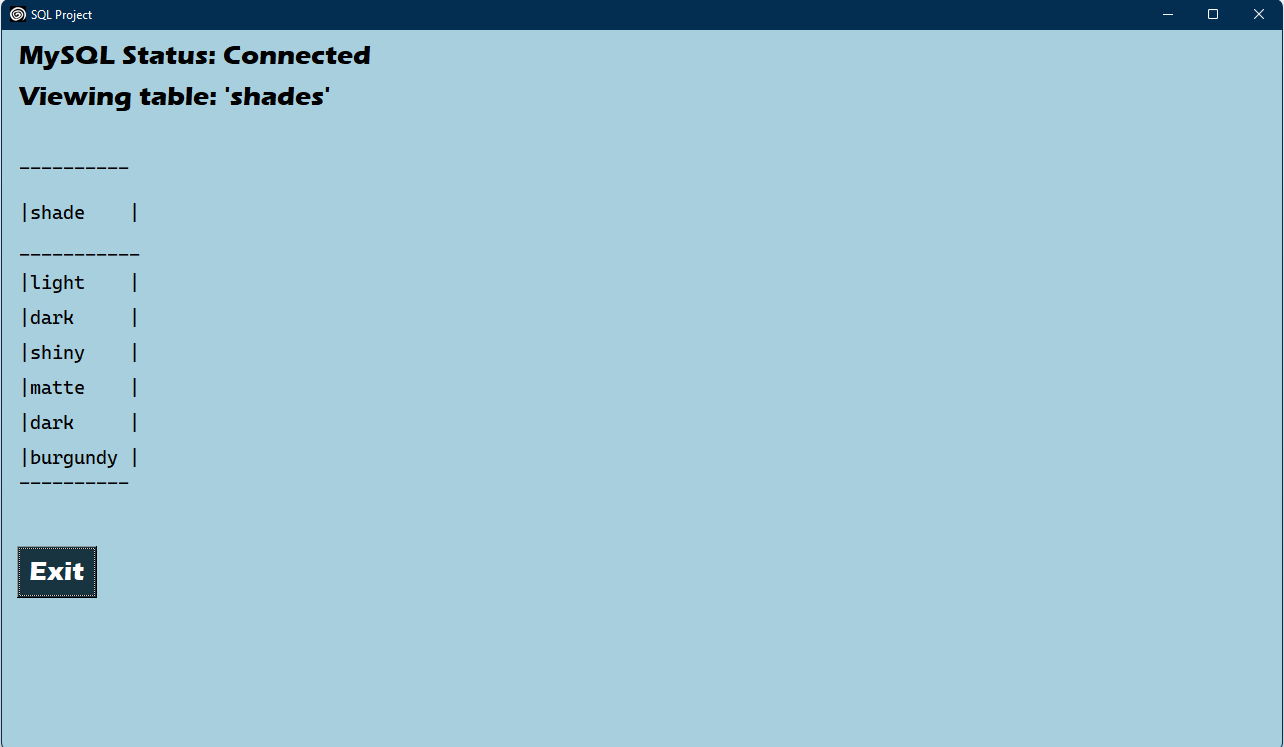


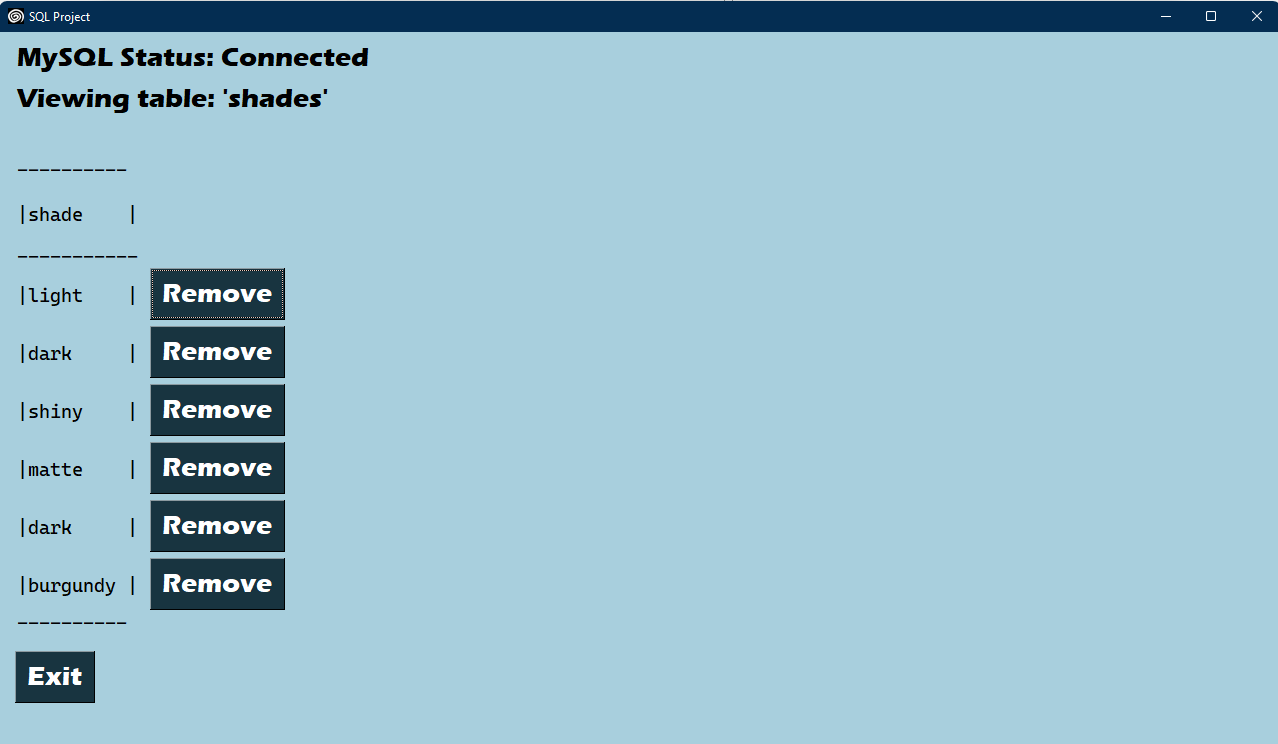


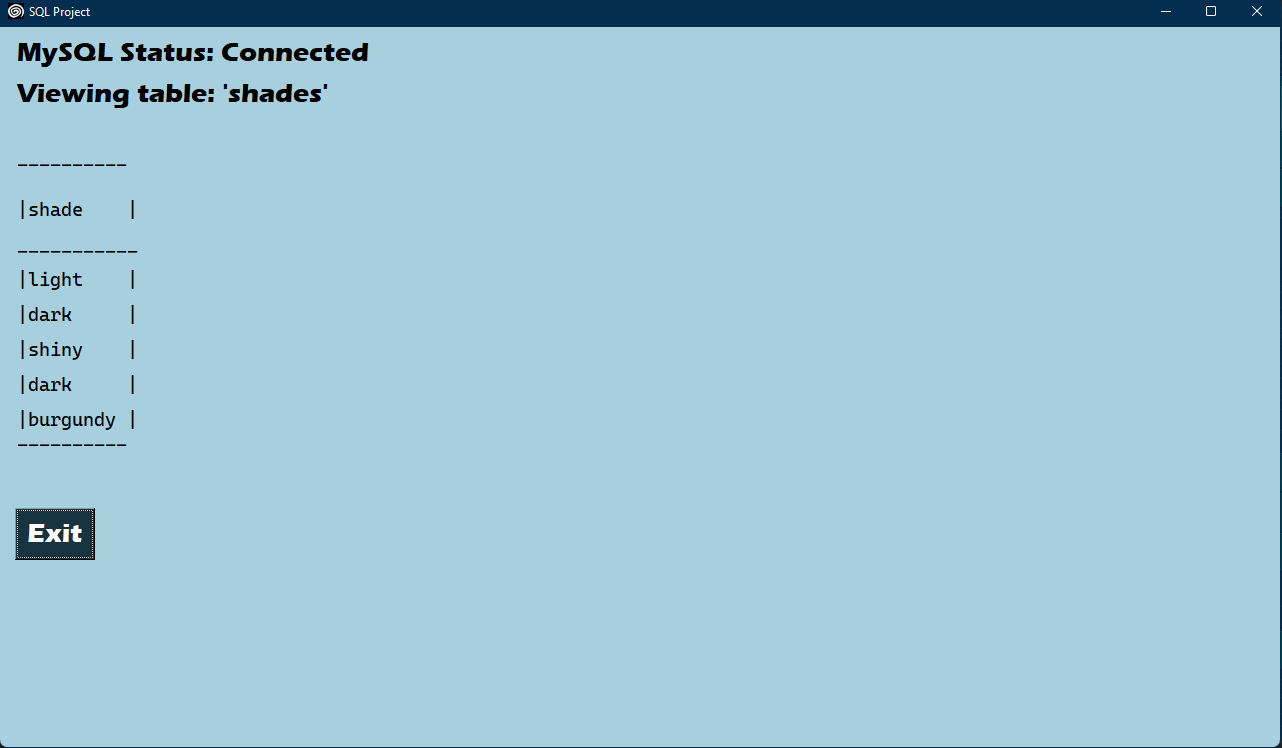


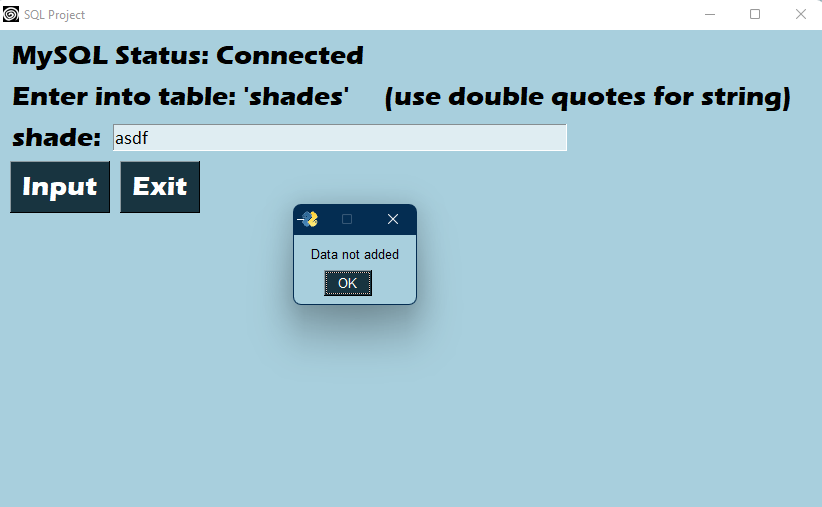
**

**

**

**

**

**

Quotes missing

**Future Enhancement of Project:**

We can add more complex argument and query capability

We can export it into a complete and standalone file.

We can also use more efficient algorithms like binary and radial sort.

We can make more complex gui elements like navbars

**Bibliography:**

Text book - Class XII – Computer Science, Preeti Arora

cbse.nic.in

https://www.geeksforgeeks.org/

https://www.w3schools.com/python/

https://en.wikipedia.org

<https://mockaroo.com>

https://www.pysimplegui.org/