Aquino,Aaron Jan O.
C204

Finals Lab Task 6.

MySQL CRUD Operations in Python Using GUI Tkinter

Step 1. Make sure you install the necessary prerequisites:

a. MySQL-Connector in Pycharm

b. Activate xampp (Apache and Mysql)

c. Create a database named: cars DB

d. Import the sql file (carsDB.sql) to load the tables and records

E. Create a user named(cs204) with password (asdf123) and assign full access to the

database - Use this credentials when connecting to the database

Step 2. See the GUI Design of the Demo interface

Step 3. Try the code below:

Get the copy of the following files and load in pycharm:

Link here:

https://drive.google.com/drive/folders/1e6Eh55qLAwepf0A_I8GKh70elW6jAxJj?usp=sharing

1. connectDb.py

2. main.py

3. window.py

Step 4. Run the program main.py (and test all the functions (CRUD)) it should be free from errors.

Make a screenshot of your output as proof that you were able to configure the program properly

Step 5. Add the ff: Functions in the GUI . Choose 1 only

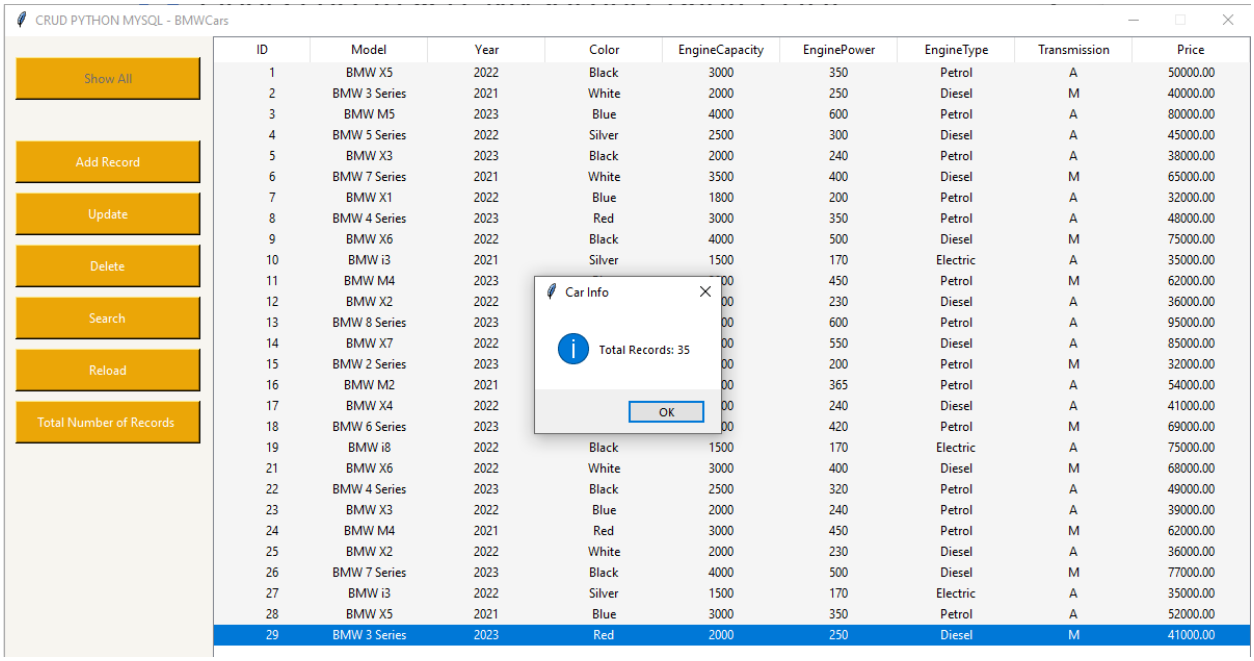
1. Insert a Label and Text widget that will display the ff: infos:

a. the total Number of Records,

b. Car Model with the Highest Price,

c. Total Number of Manual Cars

d. Total number of and Automatic Cars



ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
1	BMW X5	2022	Black	3000	350	Petrol	A	50000.00
2	BMW 3 Series	2021	White	2000	250	Diesel	M	40000.00
3	BMW M5	2023	Blue	4000	600	Petrol	A	80000.00
4	BMW 5 Series	2022	Silver	2500	300	Diesel	A	45000.00
5	BMW X3	2023	Black	2000	240	Petrol	A	38000.00
6	BMW 7 Series	2021	White	3500	400	Diesel	M	65000.00
7	BMW X1	2022	Blue	1800	200	Petrol	A	32000.00
8	BMW 4 Series	2023	Red	3000	350	Petrol	A	48000.00
9	BMW X6	2022	Black	4000	500	Diesel	M	75000.00
10	BMW i3	2021	Silver	1500	170	Electric	A	35000.00
11	BMW M4	2023		00	450	Petrol	M	62000.00
12	BMW X2	2022		00	230	Diesel	A	36000.00
13	BMW 8 Series	2023		00	600	Petrol	A	95000.00
14	BMW X7	2022		00	550	Diesel	A	85000.00
15	BMW 2 Series	2023		00	200	Petrol	M	32000.00
16	BMW M2	2021		00	365	Petrol	A	54000.00
17	BMW X4	2022		00	240	Diesel	A	41000.00
18	BMW 6 Series	2023		00	420	Petrol	M	69000.00
19	BMW i8	2022	Black	1500	170	Electric	A	75000.00
21	BMW X6	2022	White	3000	400	Diesel	M	68000.00
22	BMW 4 Series	2023	Black	2500	320	Petrol	A	49000.00
23	BMW X3	2022	Blue	2000	240	Petrol	A	39000.00
24	BMW M4	2021	Red	3000	450	Petrol	M	62000.00
25	BMW X2	2022	White	2000	230	Diesel	A	36000.00
26	BMW 7 Series	2023	Black	4000	500	Diesel	M	77000.00
27	BMW i3	2022	Silver	1500	170	Electric	A	35000.00
28	BMW X5	2021	Blue	3000	350	Petrol	A	52000.00
29	BMW 3 Series	2023	Red	2000	250	Diesel	M	41000.00

```python
#window.py


import tkinter as tk
from tkinter import font
from tkinter import ttk
from connectDB import *
from tkinter import messagebox

class Window:
    cnn = ConnectDB(host="localhost", user="c204", password="asdf123",
database="cars db")

    def __init__(self, root):
        self.root = root
        self.settings()
        self.create_widgets()

    def settings(self):
        self.root.title("CRUD PYTHON MYSQL - BMWCars")
        self.root.resizable(0, 0)

        widthScreen = self.root.winfo_screenwidth()
        heightScreen = self.root.winfo_screenheight()
        widthWindow = 1200
        heightWindow = 600
        pwidth = int(widthScreen / 2 - widthWindow / 2)
        pheight = int(heightScreen / 2 - heightWindow / 2)
        self.root.geometry(f"{widthWindow}x{heightWindow}+{pwidth}+{pheight -
30}")

    def create_widgets(self):
        # FRAME BUTTONS
        frame1 = tk.Frame(self.root, width=200, height=600, bg="#f7f5f0")
```

```
        frame1.place(x=0, y=0)

        self.buttonInit = tk.Button(frame1, text="Show All",
command=self.fnInit,
                                    width=24, height=2, background="#eba607",
                                    foreground="white")
        self.buttonInit.place(x=10, y=20)

        self.buttonNew = tk.Button(frame1, text="Add Record",
command=self.InsertData,
                                   width=24, height=2, background="#eba607",
foreground="white")
        self.buttonNew.place(x=10, y=100)

        self.buttonUpdate = tk.Button(frame1, text="Update",
command=self.UpdateData,
                                      width=24, height=2,
background="#eba607", foreground="white")
        self.buttonUpdate.place(x=10, y=150)

        self.buttonDelete = tk.Button(frame1, text="Delete",
command=self.DeleteData,
                                      width=24, height=2,
background="#eba607", foreground="white")
        self.buttonDelete.place(x=10, y=200)

        self.buttonSearch = tk.Button(frame1, text="Search",
command=self.SearchData,
                                      width=24, height=2,
background="#eba607", foreground="white")
        self.buttonSearch.place(x=10, y=250)

        self.buttonReload = tk.Button(frame1, text="Reload",
command=self.fnInit,
                                      width=24, height=2,
background="#eba607", foreground="white")
        self.buttonReload.place(x=10, y=300)

        # Step 2: Show Info Button
        self.buttonTotalInfo = tk.Button(frame1, text="Total Number of
Records ", command=self.show_info,
                                         width=24, height=2,
background="#eba607", foreground="white")
        self.buttonTotalInfo.place(x=10, y=350)

        # FRAME INPUT
        self.frame2 = tk.Frame(self.root, width=300, height=600,
bg="#CCCCCC")

        lbl1 = tk.Label(self.frame2, text="ID", background="#CCCCCC")
        lbl1.place(x=10, y=15)
        self.entry1 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry1.place(x=10, y=40)

        lbl2 = tk.Label(self.frame2, text="Model:", background="#CCCCCC")
        lbl2.place(x=10, y=80)
```

```
        self.entry2 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry2.place(x=10, y=105)

        lbl3 = tk.Label(self.frame2, text="Year Make:", background="#CCCCCC")
        lbl3.place(x=10, y=145)
        self.entry3 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry3.place(x=10, y=170)

        lbl4 = tk.Label(self.frame2, text="Color:", background="#CCCCCC")
        lbl4.place(x=10, y=210)
        self.entry4 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry4.place(x=10, y=235)

        lbl5 = tk.Label(self.frame2, text="Engine Capacity:",
background="#CCCCCC")
        lbl5.place(x=10, y=275)
        self.entry5 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry5.place(x=10, y=300)

        lbl6 = tk.Label(self.frame2, text="Engine Power:",
background="#CCCCCC")
        lbl6.place(x=10, y=340)
        self.entry6 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry6.place(x=10, y=365)

        lbl7 = tk.Label(self.frame2, text="Engine Type:",
background="#CCCCCC")
        lbl7.place(x=10, y=405)
        self.entry7 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry7.place(x=10, y=430)

        lbl8 = tk.Label(self.frame2, text="Transmission Type:",
background="#CCCCCC")
        lbl8.place(x=10, y=470)
        self.entry8 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry8.place(x=10, y=495)

        lbl9 = tk.Label(self.frame2, text="Price", background="#CCCCCC")
        lbl9.place(x=10, y=535)
        self.entry9 = tk.Entry(self.frame2, width=30,
font=font.Font(size=12))
        self.entry9.place(x=10, y=560)

        # Frame Buttons Save and Cancel
        self.buttonSave = tk.Button(frame1, text="Save", command=self.save,
                                    width=24, height=2, background="#006400",
foreground="black")

        self.buttonCancel = tk.Button(frame1, text="Cancel",
command=self.cancel,
```

```
                                        width=24, height=2,
background="#8B0000", foreground="black")

        style = ttk.Style()
        style.configure("Custom.Treeview", background="whitesmoke",
                        foreground="black")

        # Table's frame of database
        self.grid = ttk.Treeview(self.root, columns=("col1", "col2", "col3",
"col4"
                                                     , "col5", "col6",
"col7", "col8"),
                                 style="Custom.Treeview")
        self.grid.column("#0", width=50, anchor=tk.CENTER)
        for i in range(1, 9):
            self.grid.column(f"col{i}", width=70, anchor=tk.CENTER)

        self.grid.heading("#0", text="ID")
        self.grid.heading("col1", text="Model")
        self.grid.heading("col2", text="Year")
        self.grid.heading("col3", text="Color")
        self.grid.heading("col4", text="EngineCapacity")
        self.grid.heading("col5", text="EnginePower")
        self.grid.heading("col6", text="EngineType")
        self.grid.heading("col7", text="Transmission")
        self.grid.heading("col8", text="Price")

        self.grid.place(x=200, y=0, width=999, height=599)

    # ------------------ Step 3: Show Info ------------------
    def show_info(self):
        self.cnn.connect()
        data = self.cnn.execute_select("car")
        total_records = len(data)
        self.cnn.disconnect()
        messagebox.showinfo("Car Info",f"Total Records: {total_records}")

    def fnInit(self):
        self.grid.delete(*self.grid.get_children())
        self.cnn.connect()
        data = self.cnn.execute_select("car")
        for row in data:
            self.grid.insert("", tk.END, text=row[0],
                             values=row[1:])
        self.cnn.disconnect()
        self.buttonInit.config(state="disabled")

    def cancel(self):
        self.buttonSave.place_forget()
        self.buttonCancel.place_forget()
        self.grid.place(x=200, y=0, width=999, height=599)
        self.entry1.config(state="normal")
        for entry in [self.entry1,self.entry2,self.entry3,self.entry4,

self.entry5,self.entry6,self.entry7,self.entry8,self.entry9]:
            entry.delete(0, "end")
        for btn in
```

```
[self.buttonUpdate,self.buttonNew,self.buttonDelete,self.buttonSearch,self.bu
ttonReload]:
            btn.config(state="normal")

    def save(self):
        txtid = txtmodel = txtyear = txtcolor = txttype = txttrans = ""
        txtcapacity = txtpower = 0
        txtprice = 0.0

        try:
            txtid = int(self.entry1.get())
            txtmodel = self.entry2.get()
            txtyear = self.entry3.get()
            txtcolor = self.entry4.get()
            txtcapacity = int(self.entry5.get())
            txtpower = int(self.entry6.get())
            txttype = self.entry7.get()
            txttrans = self.entry8.get()
            txtprice = float(self.entry9.get())
        except ValueError:
            messagebox.showerror("Error", "All fields must be filled
correctly.")
            return
        finally:
            for entry in [self.entry1,self.entry2,self.entry3,self.entry4,

self.entry5,self.entry6,self.entry7,self.entry8,self.entry9]:
                entry.delete(0, "end")

        self.cnn.connect()
        if self.entry1.cget("state") == "normal":
            self.cnn.execute_insert("car", txtid, txtmodel, txtyear,
txtcolor,
                                    txtcapacity, txtpower, txttype, txttrans,
txtprice)
        else:
            self.cnn.execute_update("car", txtid, txtmodel, txtyear,
txtcolor,
                                    txtcapacity, txtpower, txttype, txttrans,
txtprice)
        self.cnn.disconnect()
        self.fnInit()
        self.buttonSave.place_forget()
        self.buttonCancel.place_forget()

        for btn in
[self.buttonUpdate,self.buttonNew,self.buttonDelete,self.buttonSearch,self.bu
ttonReload]:
            btn.config(state="normal")
        self.entry1.config(state="normal")

    def InsertData(self):
        self.grid.place(x=500, y=0, width=699, height=599)
        self.frame2.place(x=200, y=0)
        self.buttonSave.place(x=10, y=495)
        self.buttonCancel.place(x=10, y=545)
        for btn in
```

```
[self.buttonUpdate,self.buttonNew,self.buttonDelete,self.buttonSearch,self.bu
ttonReload]:
            btn.config(state="disabled")

    def UpdateData(self):
        selection = self.grid.selection()
        if selection:
            self.grid.place(x=500, y=0, width=699, height=599)
            self.frame2.place(x=200, y=0)
            self.buttonSave.place(x=10, y=495)
            self.buttonCancel.place(x=10, y=545)
            for btn in
[self.buttonUpdate,self.buttonNew,self.buttonDelete,self.buttonSearch,self.bu
ttonReload]:
                btn.config(state="disabled")
            id_selectioned = self.grid.item(selection)['text']
            values = self.grid.item(selection)['values']
            if values:
                for i, entry in
enumerate([self.entry2,self.entry3,self.entry4,self.entry5,self.entry6,self.e
ntry7,self.entry8,self.entry9]):
                    entry.insert(0, values[i])
                self.entry1.insert(0, id_selectioned)
                self.entry1.config(state="disabled")
        else:
            messagebox.showerror("Error", "You must select a data")

    def DeleteData(self):
        selection = self.grid.selection()
        if selection:
            id_selectioned = self.grid.item(selection)['text']
            self.cnn.connect()
            self.cnn.execute_delete("car", id_selectioned)
            self.cnn.disconnect()
            self.fnInit()

    def SearchData(self):
        new_window = tk.Toplevel(self.root)
        new_window.title("Search")
        new_window.resizable(0, 0)
        widthScreen = self.root.winfo_screenwidth()
        heightScreen = self.root.winfo_screenheight()
        widthWindow = 700
        heightWindow = 50
        pwidth = int(widthScreen / 2 - widthWindow / 2)
        pheight = int(heightScreen / 2 - heightWindow / 2)
        new_window.geometry(f"{widthWindow}x{heightWindow}+{pwidth}+{pheight
- 60}")

        def show_search_data(i, search_text):
            found_items = []
            all_items_values = []
            self.cnn.connect()
            data = self.cnn.execute_select("car")
            self.cnn.disconnect()
            all_items_values = list(data)
            for j in range(len(all_items_values)):
```

```
                if search_text.lower() ==
str(all_items_values[j][i]).lower():
                    found_items.append(all_items_values[j])
            self.grid.delete(*self.grid.get_children())
            for data in found_items:
                self.grid.insert('', tk.END, text=data[0], values=data[1:])
            new_window.destroy()

        def get_selected_option(search_text):
            selected_option = radio_var.get()
            if selected_option == "option1":
                show_search_data(0, search_text)
            elif selected_option == "option2":
                show_search_data(1, search_text)
            elif selected_option == "option3":
                show_search_data(2, search_text)
            elif selected_option == "option4":
                show_search_data(8, search_text)
            else:
                show_search_data(0, search_text)

        radio_var = tk.StringVar()
        ttk.Radiobutton(new_window, text="Id", variable=radio_var,
value="option1").place(x=30, y=12)
        ttk.Radiobutton(new_window, text="Model", variable=radio_var,
value="option2").place(x=80, y=12)
        ttk.Radiobutton(new_window, text="Year", variable=radio_var,
value="option3").place(x=160, y=12)
        ttk.Radiobutton(new_window, text="Price", variable=radio_var,
value="option4").place(x=240, y=12)
        entry_search = tk.Entry(new_window, width=30,
font=font.Font(size=10))
        entry_search.place(x=320, y=14)
        ttk.Button(new_window, text="Get Selected Option", command=lambda:
get_selected_option(entry_search.get())).place(x=550, y=11)
```

```
#connectDB.py
import mysql.connector
from tkinter import messagebox

class ConnectDB:
    def __init__(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connectDB = None

    def connect(self):
        try:
            self.connectDB = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                database=self.database,
                ssl_disabled=True
```

```
            )
            print("Successfully connected to the database!")
        except mysql.connector.Error as error:
            print("Something went wrong connecting to the database: ", error)

    def disconnect(self):
        if self.connectDB:
            self.connectDB.close()
            print("Successfully disconnected from the database!")

    def execute_insert(self, table, id, model, year, color, capacity, power,
type, transmission, price):
        sql = f"INSERT INTO {table}(id, model, year, color, engineCapacity,
enginePower, engineType, transmission, price) VALUES({id},'{model}',
'{year}', '{color}', {capacity},{power}, '{type}','{transmission}',{price})"
        self.commit_to_db(sql)

    def execute_delete(self, table, id):
        sql = f"DELETE FROM {table} WHERE id = {id}"
        self.commit_to_db(sql)

    def execute_update(self, table, id, model, year, color, capacity, power,
engineType, transmission, price):
        sql = f"UPDATE {table} SET model='{model}', year='{year}',
color='{color}', engineCapacity={capacity}, enginePower={power},
engineType='{engineType}',transmission='{transmission}', price={price} WHERE
id={id}"
        self.commit_to_db(sql)

    def commit_to_db(self, sql):
        cursor = self.connectDB.cursor()
        try:
            cursor.execute(sql)
            self.connectDB.commit()
            print("Query successfully executed")
            messagebox.showinfo("Successfully", "Query successfully executed.
Good Work!")
        except mysql.connector.Error as error:
            self.connectDB.rollback()
            print("Error executing the query:", error)
            messagebox.showerror("Error", "Duplicate ID entry or invalid
input, please try again!")

    def execute_select(self, table):
        sql = f"SELECT * FROM {table}"
        cursor = self.connectDB.cursor()
        try:
            cursor.execute(sql)
            rows = cursor.fetchall()
            return rows
        except mysql.connector.Error as error:
            print("Error executing the query:", error)
            return []

    # Step 1: Total records method
    def get_total_records(self):
        cursor = self.connectDB.cursor()
```

```
        cursor.execute("SELECT COUNT(*) FROM car")
        result = cursor.fetchone()
        return result[0] if result else 0

    def __str__(self):
        data = self.execute_select("car")
        aux = ""
        for row in data:
            aux += str(row) + "\n"
        return aux
```

```
#main.py

import tkinter as tk
import window

def main():
    root = tk.Tk()
    crud = window.Window(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```