

Finals Lab Task 6. Mysql and Tkinter HUI

Problem:

Finals Lab Task 6. MySQL CRUD Operations in Python Using GUI Tkinter

Step 1. Make sure you install the necessary prerequisites:

- MySQL-Connector in Pycharm
- Activate xampp (Apache and Mysql)
- Create a database named: cars DB
- Import the sql file (carsDB.sql) to load the tables and records
- 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

The screenshot shows a Tkinter application window titled 'Cars Database'. On the left is a vertical menu bar with buttons for 'Home', 'Add Record', 'View Record', 'Search', and 'Exit'. The main area contains a table with the following data:

ID	Model	Year	Color	EnginCapacity	EnginPower	EnginType	Transmission	Price
1	BMW 11	2012	Black	2000	150	Diesel	A	30000.00
2	BMW 2 Series	2017	White	2000	200	Diesel	M	40000.00
3	BMW 3 Series	2016	Blue	2000	200	Diesel	A	40000.00
4	BMW 5 Series	2012	Grey	2000	200	Diesel	A	40000.00
5	BMW 6 Series	2010	Black	2000	200	Petrol	A	50000.00
6	BMW 7 Series	2015	White	2000	200	Diesel	M	50000.00
7	BMW X1	2012	Blue	1600	120	Petrol	A	20000.00
8	BMW X2 Series	2012	Red	2000	200	Petrol	A	40000.00
9	BMW X3	2012	Black	2000	200	Diesel	M	70000.00
10	BMW X4	2017	Grey	2000	200	Diesel	A	50000.00
11	BMW X5	2012	Blue	2000	200	Petrol	M	50000.00
12	BMW X6	2012	White	2000	200	Diesel	A	60000.00
13	BMW 1 Series	2013	Black	1600	100	Petrol	A	15000.00
14	BMW 27	2012	Grey	1600	120	Diesel	A	15000.00
15	BMW 2 Series	2012	Black	1600	120	Petrol	M	20000.00
16	BMW 320d	2017	White	2000	180	Petrol	A	30000.00
17	BMW 320i	2012	Blue	2000	200	Diesel	A	40000.00
18	BMW 520d	2012	Black	2000	180	Petrol	M	50000.00
19	BMW 520i	2012	Red	2000	200	Petrol	M	60000.00
20	BMW 620d	2012	Black	2000	200	Petrol	A	70000.00
21	BMW 620i	2012	White	2000	200	Diesel	M	60000.00
22	BMW 720d	2017	Black	2000	200	Petrol	A	40000.00
23	BMW 720i	2012	Blue	2000	200	Petrol	A	50000.00
24	BMW 320i	2017	Red	2000	200	Petrol	M	20000.00
25	BMW 320d	2012	White	2000	180	Diesel	A	30000.00
26	BMW 520d	2012	Black	2000	180	Diesel	M	50000.00
27	BMW 520i	2012	Grey	2000	200	Diesel	A	40000.00
28	BMW 318i	2017	Red	2000	180	Petrol	M	20000.00

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_l8GKh70elW6jAxJ?usp=sharing

- connectDb.py
- main.py
- 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:

- the total Number of Records,
- Car Model with the Highest Price,
- Total Number of Manual Cars
- Total number of and Automatic Cars

Sample Code:

The screenshot shows a PyCharm IDE window with four tabs at the top: 'main.py', 'connectDB.py', 'carsDB.sql', and 'window.py'. The 'main.py' tab is active. The code in 'main.py' is as follows:

```
1 import tkinter as tk
2 import window
3
4 1 usage
5 def main():
6     root = tk.Tk()
7     crud = window.Window(root)
8     root.mainloop()
9
10 if __name__ == "__main__":
11     main()
```

The code uses Tkinter for the GUI and imports a 'window' module. It defines a main function that creates a Tk root window, creates an instance of the Window class, and starts the main loop. A conditional statement checks if the script is run directly to call the main function. There are several code analysis annotations: a yellow lightbulb icon on line 10, a red exclamation mark icon on line 4, and a green checkmark icon on line 9.

The screenshot shows a code editor with several tabs at the top: main.py, connectDB.py, carsDB.sql, and window.py. The main.py tab is active and displays the following Python code:

```
main.py
1 import mysql.connector
2 from tkinter import messagebox
3
4 1 usage
5 class ConnectDB:
6     def __init__(self, host, user, password, database):
7         self.host = host
8         self.user = user
9         self.password = password
10        self.database = database
11        self.connectDB = None
12
13    5 usages
14    def connect(self):
15        try:
16            self.connectDB = mysql.connector.connect(
17                host=self.host,
18                user=self.user,
19                password=self.password,
20                database=self.database,
21                ssl_disabled=True
22            )
23            print("Successfully connected to the database!")
24        except mysql.connector.Error as error:
25            print("Something went wrong connecting to the database: ", error)
26
27    5 usages
28    def disconnect(self):
29        if self.connectDB:
30            self.connectDB.close()
31            print("Successfully disconnected from the database!")
32
33 1 usage
34    def execute_insert(self, table, id, model, year, color, capacity, power, type, transmission, price):
35        sql = f"INSERT INTO {table}({id}, {model}, {year}, {color}, {capacity}, {power}, {type}, {transmission}, {price}) VALUES({id}, '{model}', '{year}', '{color}', '{capacity}', '{power}', '{type}', '{transmission}', '{price}')"
36        self.commit_to_db(sql)
```

```
43     def execute_update(self, table, id, model, year, color, capacity, power, engineType, transmission, price):
44         sql = f"UPDATE {table} SET model={model}, year={year}, color={color}, engineCapacity={capacity}, enginePower={power}, engineType='{engineType}', transmission='{transmission}', price={price} WHERE id={id}"
45         self.commit_to_db(sql)
46
47     3 usages:
48     def commit_to_db(self, sql):
49         cursor = self.connectDB.cursor()
50         try:
51             cursor.execute(sql)
52             self.connectDB.commit()
53             print("Query successfully executed")
54             messagebox.showinfo(title="Successfully", message="Query successfully executed. Good Work!")
55         except mysql.connector.Error as error:
56             self.connectDB.rollback()
57             print("Error executing the query:", error)
58             messagebox.showerror(title="Error", message="Duplicate ID entry or invalid input, please try again!")
59
60     4 usages:
61     def execute_select(self, table):
62         sql = f"SELECT * FROM {table}"
63         cursor = self.connectDB.cursor()
64         try:
65             cursor.execute(sql)
66             rows = cursor.fetchall()
67             return rows
68         except mysql.connector.Error as error:
69             print("Error executing the query:", error)
70             return []
71
72     1 usage:
73     def __str__(self):
74         data = self.execute_select("car")
75         aux = ""
76         for row in data:
77             aux += str(row) + "\n"
78
79     1 usage:
```

```
65     def get_total_records(self):
66         cursor = self.connectDB.cursor()
67         cursor.execute("SELECT COUNT(*) FROM car")
68         result = cursor.fetchone()
69         return result[0] if result else 0
70
71     1 usage:
72     def __str__(self):
73         data = self.execute_select("car")
74         aux = ""
75         for row in data:
76             aux += str(row) + "\n"
77
78     1 usage:
```

```
main.py × connectDB.py × carsDB.sql × window.py ×
0 results ↑ ↓ + - E Y

1 import tkinter as tk
2 from tkinter import font
3 from tkinter import ttk
4 from connectDB import *
5 from tkinter import messagebox
6
7
8     1 usage
9 class Window:
10     cnn = ConnectDB(host="localhost", user="cs204", password="asdf123", database="cars db")
11
12     def __init__(self, root):
13         self.root = root
14         self.settings()
15         self.create_widgets()
16
17     1 usage
18     def settings(self):
19         self.root.title("CRUD PYTHON MYSQL - BMWcars")
20         self.root.resizable(0, 0)
21         widthScreen = self.root.winfo_screenwidth()
22         heightScreen = self.root.winfo_screenheight()
23         widthWindow = 1200
24         heightWindow = 600
25         pwidht = int(widthScreen / 2 - widthWindow / 2)
26         pheight = int(heightScreen / 2 - heightWindow / 2)
27         self.root.geometry(f"{widthWindow}x{heightWindow}+{pwidht}+{pheight - 30}")
28
29     1 usage
30     def create_widgets(self):
31         # FRAME BUTTONS
32         frame1 = tk.Frame(self.root, width=200, height=600, bg="#f7f5f0")
33         frame1.place(x=0, y=0)
34
35         self.buttonInit = tk.Button(frame1, text="Show All", command=self.fnInit,
36                                     width=24, height=2, background="#eba607", foreground="white")
37         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)

# Button to show total records
self.buttonTotalInfo = tk.Button(frame1, text="Show Total Records", command=self.show_info,
                           width=24, height=2, background="#0066cc", foreground="white")
self.buttonTotalInfo.place(x=10, y=350)

# Label to display total records
self.labelTotalRecords = tk.Label(frame1, text="Total Records: 0", background="#f7f5f0", font=("Arial", 12))
self.labelTotalRecords.place(x=10, y=400)
```

```

66     self.frame2 = tk.Frame(self.root, width=300, height=600, bg="#CCCCCC")
67
68     lbl1 = tk.Label(self.frame2, text="ID", background="#CCCCCC")
69     lbl1.place(x=10, y=15)
70     self.entry1 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
71     self.entry1.place(x=10, y=40)
72
73     lbl2 = tk.Label(self.frame2, text="Model:", background="#CCCCCC")
74     lbl2.place(x=10, y=80)
75     self.entry2 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
76     self.entry2.place(x=10, y=105)
77
78     lbl3 = tk.Label(self.frame2, text="Year Make:", background="#CCCCCC")
79     lbl3.place(x=10, y=145)
80     self.entry3 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
81     self.entry3.place(x=10, y=170)
82
83     lbl4 = tk.Label(self.frame2, text="Color:", background="#CCCCCC")
84     lbl4.place(x=10, y=210)
85     self.entry4 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
86     self.entry4.place(x=10, y=235)
87
88     lbl5 = tk.Label(self.frame2, text="Engine Capacity:", background="#CCCCCC")
89     lbl5.place(x=10, y=275)
90     self.entry5 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
91     self.entry5.place(x=10, y=300)
92
93     lbl6 = tk.Label(self.frame2, text="Engne Motor:", background="#CCCCCC")
94     lbl6.place(x=10, y=340)
95     self.entry6 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
96     self.entry6.place(x=10, y=365)
97
98     lbl7 = tk.Label(self.frame2, text="Engine Type:", background="#CCCCCC")
99     lbl7.place(x=10, y=405)
100    self.entry7 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
101    self.entry7.place(x=10, y=430)

```

```

105    lbl8 = tk.Label(self.frame2, text="Transmission Type:", background="#CCCCCC")
106    lbl8.place(x=10, y=470)
107    self.entry8 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
108    self.entry8.place(x=10, y=495)
109
110    lbl9 = tk.Label(self.frame2, text="Price", background="#CCCCCC")
111    lbl9.place(x=10, y=535)
112    self.entry9 = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
113    self.entry9.place(x=10, y=560)
114
115    self.buttonSave = tk.Button(frame1, text="Save", command=self.save,
116                               width=24, height=2, background="#006400", foreground="black")
117    self.buttonCancel = tk.Button(frame1, text="Cancel", command=self.cancel,
118                               width=24, height=2, background="#8B0000", foreground="black")
119
120    style = ttk.Style()
121    style.configure("Custom.Treeview", background="whitesmoke", foreground="black")
122
123    # Table to display database records
124    self.grid = ttk.Treeview(self.root, columns=( "col1", "col2", "col3", "col4",
125                                                "col5", "col6", "col7", "col8"),
126                                style="Custom.Treeview")
127    self.grid.column("#0", width=50, anchor=tk.CENTER)
128    for i in range(1, 9):
129        self.grid.column(f"col{i}", width=70, anchor=tk.CENTER)
130    self.grid.heading("#0", text="ID")
131    self.grid.heading("col1", text="Model")
132    self.grid.heading("col2", text="Year")
133    self.grid.heading("col3", text="Color")
134    self.grid.heading("col4", text="EngineCapacity")
135    self.grid.heading("col5", text="EnginePower")
136    self.grid.heading("col6", text="EngineType")
137    self.grid.heading("col7", text="Transmission")
138    self.grid.heading("col8", text="Price")
139    self.grid.place(x=200, y=0, width=999, height=599)

```

```
140     def show_info(self):
141         self.cnn.connect()
142         data = self.cnn.execute_select("car")
143         total_records = len(data)
144         self.cnn.disconnect()
145         self.labelTotalRecords.config(text="Total Records: {total_records + 1}")
146
147     4 usages
148     def fnInit(self):
149         self.grid.delete(*self.grid.get_children())
150         self.cnn.connect()
151         data = self.cnn.execute_select("car")
152         for row in data:
153             self.grid.insert(parent "", tk.END, text=row[0],
154                             values=(row[1], row[2], row[3], row[4], row[5], row[6], row[7], row[8]))
155         self.cnn.disconnect()
156         self.buttonInit.config(state="disabled")
157
158     2 usages
159     def cancel(self):
160         self.buttonSave.place_forget()
161         self.buttonCancel.place_forget()
162         self.grid.place_forget()
163         self.grid.place(x=200, y=0, width=999, height=599)
164         for entry in [self.entry1, self.entry2, self.entry3, self.entry4, self.entry5,
165                       self.entry6, self.entry7, self.entry8, self.entry9]:
166             entry.config(state="normal")
167             entry.delete(first=0, last="end")
168         for button in [self.buttonUpdate, self.buttonNew, self.buttonDelete, self.buttonSearch, self.buttonReload]:
169             button.config(state="normal")
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195     1 usage
196     def InsertData(self):
197         self.grid.place(x=500, y=0, width=699, height=599)
198         self.frame2.place(x=200, y=0)
199         self.buttonSave.place(x=10, y=495)
200         self.buttonCancel.place(x=10, y=545)
201         for button in [self.buttonUpdate, self.buttonNew, self.buttonDelete, self.buttonSearch, self.buttonReload]:
202             button.config(state="disabled")
```

```
169     def save(self):
170         try:
171             txtid = int(self.entry1.get())
172             txtmodel = self.entry2.get()
173             txtyear = self.entry3.get()
174             txtcolor = self.entry4.get()
175             txtcapacity = int(self.entry5.get())
176             txtpower = int(self.entry6.get())
177             txttype = self.entry7.get()
178             txttrans = self.entry8.get()
179             txtprice = float(self.entry9.get())
180         except ValueError:
181             messagebox.showerror(title="Error", message="All fields must be filled correctly")
182             return
183
184         self.cnn.connect()
185         if self.entry1.cget("state") == "normal":
186             self.cnn.execute_insert(table="car", txtid, txtmodel, txtyear, txtcolor,
187                                    txtcapacity, txtpower, txttype, txttrans, txtprice)
188         else:
189             self.cnn.execute_update(table="car", txtid, txtmodel, txtyear, txtcolor,
190                                    txtcapacity, txtpower, txttype, txttrans, txtprice)
191         self.cnn.disconnect()
192         self.fnInit()
193         self.cancel()
194
195     1 usage
196     def InsertData(self):
197         self.grid.place(x=500, y=0, width=699, height=599)
198         self.frame2.place(x=200, y=0)
199         self.buttonSave.place(x=10, y=495)
200         self.buttonCancel.place(x=10, y=545)
201         for button in [self.buttonUpdate, self.buttonNew, self.buttonDelete, self.buttonSearch, self.buttonReload]:
202             button.config(state="disabled")
```

```

203     def UpdateData(self):
204         selection = self.grid.selection()
205         if not selection:
206             messagebox.showerror( title= "Error", message= "You must select a data")
207             return
208         self.grid.place(x=500, y=0, width=699, height=599)
209         self.frame2.place(x=200, y=0)
210         self.buttonSave.place(x=10, y=495)
211         self.buttonCancel.place(x=10, y=545)
212         for button in [self.buttonUpdate, self.buttonNew, self.buttonDelete, self.buttonSearch, self.buttonReload]:
213             button.config(state="disabled")
214
215         id_selectioned = self.grid.item(selection)['text']
216         values = self.grid.item(selection)['values']
217         entries = [self.entry1, self.entry2, self.entry3, self.entry4, self.entry5,
218                    self.entry6, self.entry7, self.entry8, self.entry9]
219         for entry, value in zip(entries, [id_selectioned] + list(values)):
220             entry.config(state="normal")
221             entry.delete(0, "end")
222             entry.insert(0, value)
223             self.entry1.config(state="disabled")
224
225         1 usage
226         def DeleteData(self):
227             selection = self.grid.selection()
228             if selection:
229                 id_selectioned = self.grid.item(selection)['text']
230                 self.cnn.connect()
231                 self.cnn.execute_delete( table: "car", id_selectioned)
232                 self.cnn.disconnect()
233                 self.fnInit()

```

```

234     def searchData(self):
235         new_window = tk.Toplevel(self.root)
236         new_window.title("Search")
237         new_window.resizable( width: 0, height: 0)
238         widthScreen = self.root.winfo_screenwidth()
239         heightScreen = self.root.winfo_screenheight()
240         widthWindow = 700
241         heightWindow = 50
242         pwidth = int(widthScreen / 2 - widthWindow / 2)
243         pheight = int(heightScreen / 2 - heightWindow / 2)
244         new_window.geometry(f"{widthWindow}x{heightWindow}+{pwidth}+{pheight - 60}")
245
246         def show_search_data(i, search_text):
247             found_items = []
248             all_items_values = []
249             self.cnn.connect()
250             data = self.cnn.execute_select("car")
251             for row in data:
252                 all_items_values.append(list(row))
253             self.cnn.disconnect()
254             for j in range(len(all_items_values)):
255                 if search_text.lower() == str(all_items_values[j][i]).lower():
256                     found_items.append(all_items_values[j])
257             self.grid.delete(*self.grid.get_children())
258             for data in found_items:
259                 self.grid.insert( parent: '', tk.END, text=data[0], values=data[1:])
260             new_window.destroy()

```

```

262     def get_selected_option(search_text):
263         selected_option = radio_var.get()
264         if selected_option == "option1":
265             show_search_data( 0, search_text)
266         elif selected_option == "option2":
267             show_search_data( 1, search_text)
268         elif selected_option == "option3":
269             show_search_data( 2, search_text)
270         elif selected_option == "option4":
271             show_search_data( 8, search_text)
272         else:
273             show_search_data( 0, search_text)
274
275
276         radio_var = tk.StringVar()
277         radio_button1 = ttk.Radiobutton(new_window, text="Id", variable=radio_var,
278                                         value="option1")
279         radio_button1.place(x=30, y=12)
280         radio_button2 = ttk.Radiobutton(new_window, text="Model", variable=radio_var,
281                                         value="option2")
282         radio_button2.place(x=80, y=12)
283         radio_button3 = ttk.Radiobutton(new_window, text="Year", variable=radio_var,
284                                         value="option3")
285         radio_button3.place(x=160, y=12)
286         radio_button4 = ttk.Radiobutton(new_window, text="Price", variable=radio_var,
287                                         value="option4")
288         radio_button4.place(x=240, y=12)
289
290         entry_search = tk.Entry(new_window, width=30, font=font.Font(size=10))
291         entry_search.place(x=320, y=14)
292         button_get_selected = ttk.Button(new_window, text="Get Selected Option",
293                                         command=lambda: get_selected_option(entry_search.get()))
294         button_get_selected.place(x=550, y=11)

```

Sample Output:

CRUD PYTHON MYSQL - BMWCars

	ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
Show All	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
Add Record	10	BMW i3	2021	Silver	1500	170	Electric	A	35000.00
	11	BMW M4	2023	Blue	3000	450	Petrol	M	62000.00
	12	BMW X2	2022	White	2000	230	Diesel	A	36000.00
Update	13	BMW 8 Series	2023	Black	4400	600	Petrol	A	95000.00
	14	BMW X7	2022	Silver	4500	550	Diesel	A	85000.00
Delete	15	BMW 2 Series	2023	Black	1800	200	Petrol	M	32000.00
	16	BMW M2	2021	White	3000	365	Petrol	A	54000.00
	17	BMW X4	2022	Blue	2000	240	Diesel	A	41000.00
Search	18	BMW 6 Series	2023	Red	3500	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
Reload	22	BMW 4 Series	2023	Black	2500	320	Petrol	A	49000.00
	23	BMW X3	2022	Blue	2000	240	Petrol	A	39000.00
Show Total Records	24	BMW M4	2021	Red	3000	450	Petrol	M	62000.00
Total Records: 35	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
	30	BMW M5	2022	White	4000	600	Petrol	A	82000.00
	31	BMW X1	2023	Black	1800	200	Petrol	A	32000.00
	32	BMW 5 Series	2021	Silver	2500	300	Diesel	A	47000.00
	33	BMW X7	2022	Black	4500	550	Diesel	A	87000.00
	34	BMW 2 Series	2023	Blue	1800	200	Petrol	M	34000.00
	35	BMW M2	2022	Red	3000	365	Petrol	A	55000.00