TAYTING, JUSTINE JAY D.
BSCS – C204

OOP
Dec. 06, 2025

Mysql and Tkinter HUI
Final Task 6

This task requires you to perform MySQL CRUD operations in Python using a Tkinter GUI. You must first install all prerequisites, including MySQL-Connector in PyCharm, activate XAMPP services, create a database named **carsDB**, import the provided SQL file, and set up a user account (**cs204 / asdf123**) with full access. After reviewing the demo GUI design, download the three project files—**connectDb.py**, **main.py**, and **window.py**—and run **main.py** to ensure all CRUD functions work without errors, taking a screenshot as proof. Finally, you must enhance the GUI by adding **one** new feature that displays information such as the total number of records, the car model with the highest price, the total number of manual cars, or the total number of automatic cars.

Code:

main.py

```
import tkinter as tk
import window

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

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

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
            )
        except mysql.connector.Error as error:
            print("Error connecting:", error)

    def disconnect(self):
```

```
        if self.connectDB:
            self.connectDB.close()

    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()
            messagebox.showinfo("Success", "Query executed successfully.")
        except mysql.connector.Error:
            self.connectDB.rollback()
            messagebox.showerror("Error", "Duplicate or invalid entry.")

    def execute_select(self, table):
        cursor = self.connectDB.cursor()
        cursor.execute(f"SELECT * FROM {table}")
        return cursor.fetchall()

    def count_all(self, table):
        cursor = self.connectDB.cursor()
        cursor.execute(f"SELECT COUNT(*) FROM {table}")
        return cursor.fetchone()[0]

    def count_automatic(self, table):
        cursor = self.connectDB.cursor()
        cursor.execute(f"SELECT COUNT(*) FROM {table} WHERE
transmission='A'")
        return cursor.fetchone()[0]

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

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="root", password="",
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):
        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, bg="#eba607", fg="white")
        self.buttonInit.place(x=10, y=20)

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

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

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

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

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

```
        self.buttonTotalCars = tk.Button(frame1, text="Total Cars",
command=self.total_cars, width=24, height=2, bg="#eba607", fg="white")
        self.buttonTotalCars.place(x=10, y=350)

        self.buttonAutomaticCars = tk.Button(frame1, text="Automatic Cars",
command=self.total_automatic, width=24, height=2, bg="#eba607", fg="white")
        self.buttonAutomaticCars.place(x=10, y=400)

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

        labels = ["ID", "Model:", "Year Make:", "Color:", "Engine Capacity:",
"Engine Motor:", "Engine Type:", "Transmission Type:", "Price"]
        self.entries = []

        y = 15
        for text in labels:
            lbl = tk.Label(self.frame2, text=text, background="#CCCCCC")
            lbl.place(x=10, y=y)
            entry = tk.Entry(self.frame2, width=30, font=font.Font(size=12))
            entry.place(x=10, y=y+25)
            self.entries.append(entry)
            y += 65

        self.entry1, self.entry2, self.entry3, self.entry4, self.entry5,
self.entry6, self.entry7, self.entry8, self.entry9 = self.entries

        self.buttonSave = tk.Button(frame1, text="Save", command=self.save,
width=24, height=2, bg="#006400", fg="black")
        self.buttonCancel = tk.Button(frame1, text="Cancel",
command=self.cancel, width=24, height=2, bg="#8B0000", fg="black")

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

        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=100, anchor=tk.CENTER)

        self.grid.heading("#0", text="ID")
        headers =
["Model","Year","Color","EngineCap","EnginePower","EngineType","Transmission"
,"Price"]
        for i, text in enumerate(headers, start=1):
            self.grid.heading(f"col{i}", text=text)

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

    def fnInit(self):
        self.grid.delete(*self.grid.get_children())
        self.cnn.connect()
        data = self.cnn.execute_select("car")
        for row in data:
```

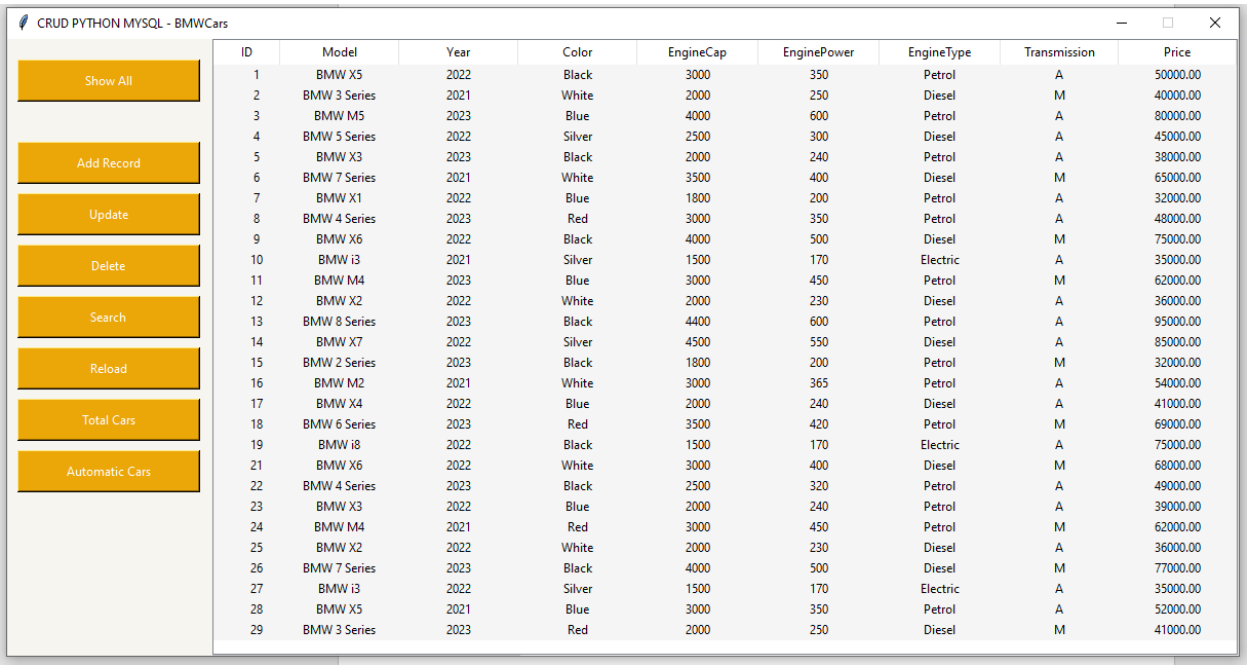
```
            self.grid.insert("", tk.END, text=row[0], values=row[1:])
        self.cnn.disconnect()

    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 e in self.entries:
            e.delete("0", "end")
        self.buttonUpdate.config(state="normal")
        self.buttonNew.config(state="normal")
        self.buttonDelete.config(state="normal")
        self.buttonSearch.config(state="normal")
        self.buttonReload.config(state="normal")

    def save(self):
        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 valid and
filled.")
            return

        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.cancel()

    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=500)
        self.buttonCancel.place(x=10, y=550)
        self.disable_actions()

    def disable_actions(self):
        self.buttonUpdate.config(state="disabled")
        self.buttonNew.config(state="disabled")
        self.buttonDelete.config(state="disabled")
        self.buttonSearch.config(state="disabled")
        self.buttonReload.config(state="disabled")
```

```
    def UpdateData(self):
        selection = self.grid.selection()
        if not selection:
            messagebox.showerror("Error", "Select a row.")
            return

        self.grid.place(x=500, y=0, width=699, height=599)
        self.frame2.place(x=200, y=0)
        self.buttonSave.place(x=10, y=500)
        self.buttonCancel.place(x=10, y=550)
        self.disable_actions()

        item = self.grid.item(selection)
        id_sel = item['text']
        values = item['values']
        entries = [id_sel] + list(values)

        for entry, data in zip(self.entries, entries):
            entry.insert(0, data)

        self.entry1.config(state="disabled")

    def DeleteData(self):
        selection = self.grid.selection()
        if not selection:
            messagebox.showerror("Error", "Select a row to delete.")
            return

        id_sel = self.grid.item(selection)['text']
        self.cnn.connect()
        self.cnn.execute_delete("car", id_sel)
        self.cnn.disconnect()
        self.fnInit()

    def total_cars(self):
        self.cnn.connect()
        total = self.cnn.count_all("car")
        self.cnn.disconnect()
        messagebox.showinfo("Total Cars", f"Total cars: {total}")

    def total_automatic(self):
        self.cnn.connect()
        total = self.cnn.count_automatic("car")
        self.cnn.disconnect()
        messagebox.showinfo("Automatic Cars", f"Automatic cars: {total}")

    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}")

        radio_var = tk.StringVar()
        options =
[("Id","option1"),("Model","option2"),("Year","option3"),("Price","option4")]
        x = 30
        for text, val in options:
            ttk.Radiobutton(new_window, text=text, variable=radio_var,
value=val).place(x=x, y=12)
            x += 70

        entry_search = tk.Entry(new_window, width=30)
        entry_search.place(x=320, y=14)

        def show_search_data(i, search_text):
            self.cnn.connect()
            data = self.cnn.execute_select("car")
            self.cnn.disconnect()
            found = [row for row in data if str(row[i]).lower() ==
search_text.lower()]
            self.grid.delete(*self.grid.get_children())
            for row in found:
                self.grid.insert("", tk.END, text=row[0], values=row[1:])
            new_window.destroy()

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

        ttk.Button(new_window, text="Search", command=lambda:
get_selected_option(entry_search.get())).place(x=550, y=11)
```

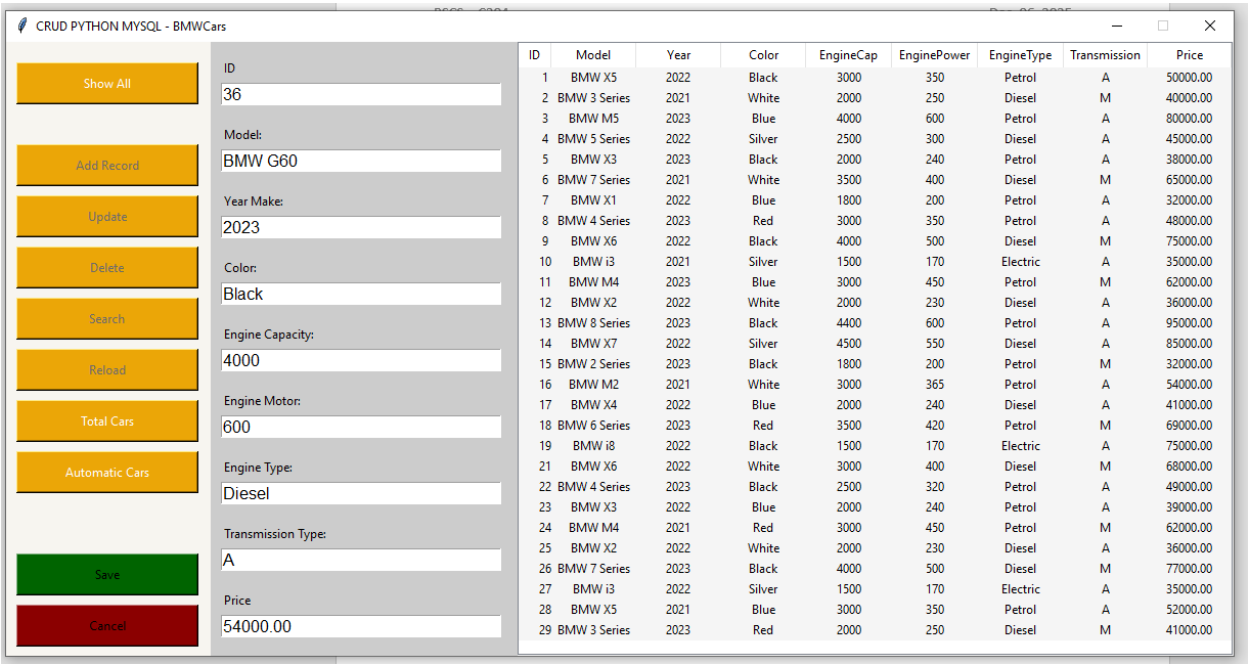
Output:

SHOW ALL



ID	Model	Year	Color	EngineCap	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	Blue	3000	450	Petrol	M	62000.00
12	BMW X2	2022	White	2000	230	Diesel	A	36000.00
13	BMW 8 Series	2023	Black	4400	600	Petrol	A	95000.00
14	BMW X7	2022	Silver	4500	550	Diesel	A	85000.00
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
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
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

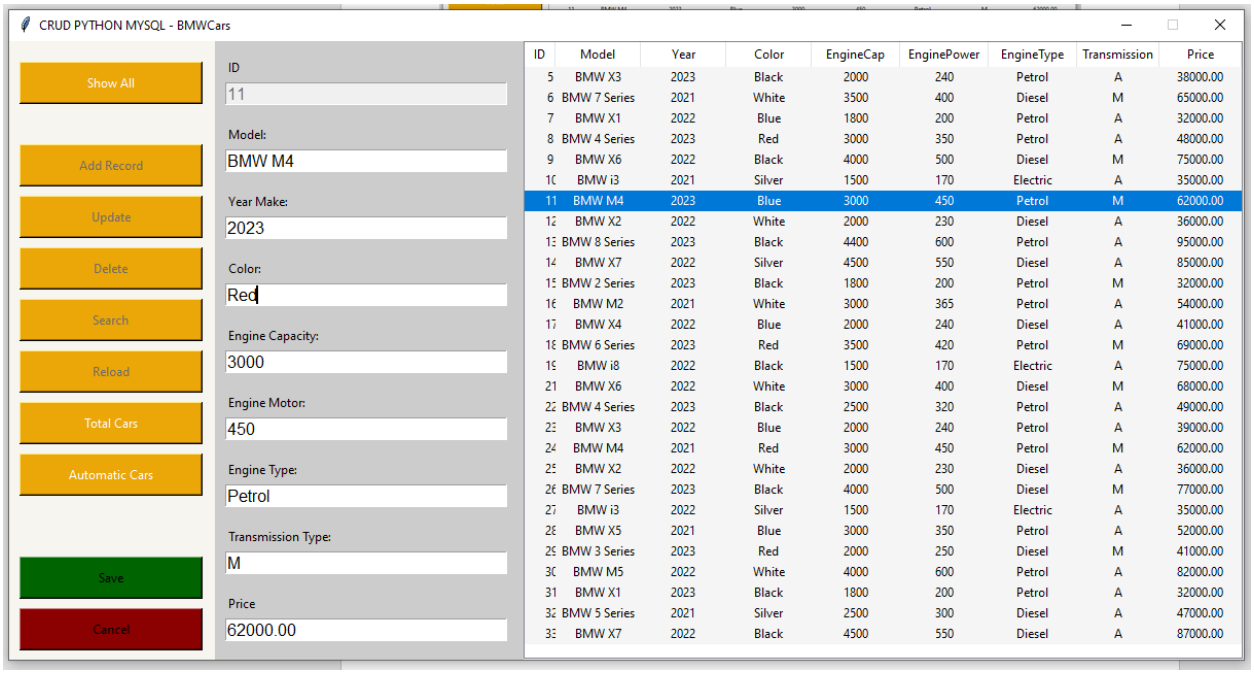
ADD RECORD



ID	Model	Year	Color	EngineCap	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	Blue	3000	450	Petrol	M	62000.00
12	BMW X2	2022	White	2000	230	Diesel	A	36000.00
13	BMW 8 Series	2023	Black	4400	600	Petrol	A	95000.00
14	BMW X7	2022	Silver	4500	550	Diesel	A	85000.00
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
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
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
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
36	BMW G60	2023	Black	4000	600	Diesel	A	54000.00

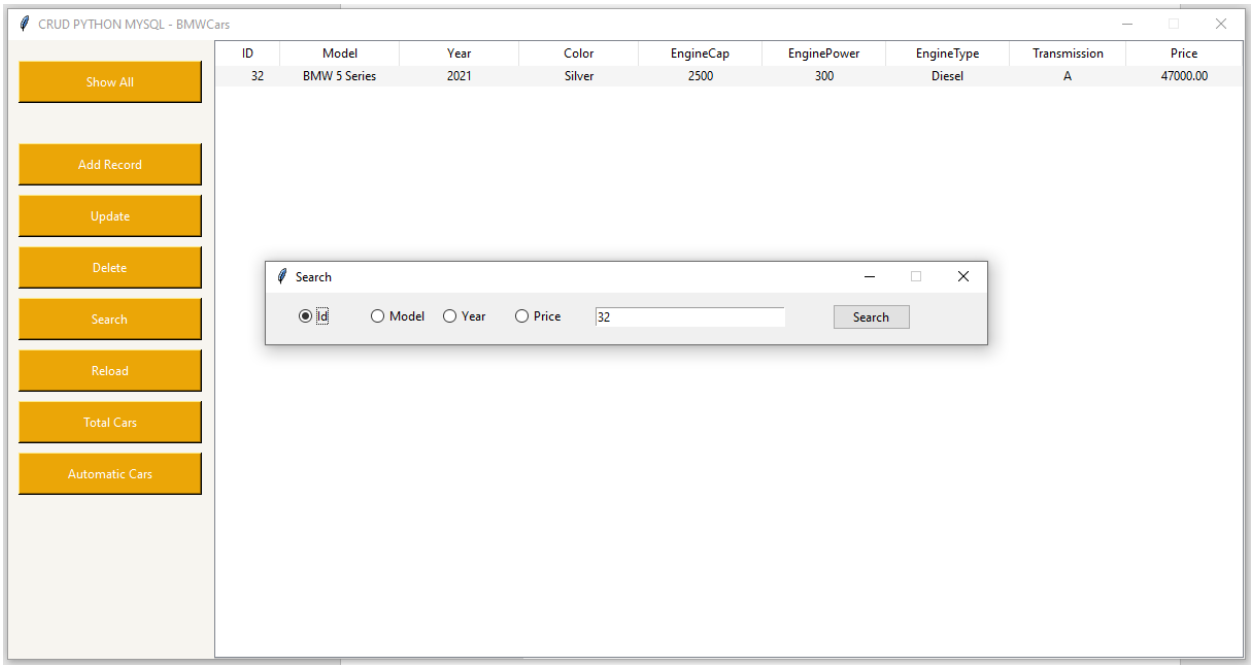
TAYTING, JUSTINE JAY D.
BSCS – C204

UPDATE



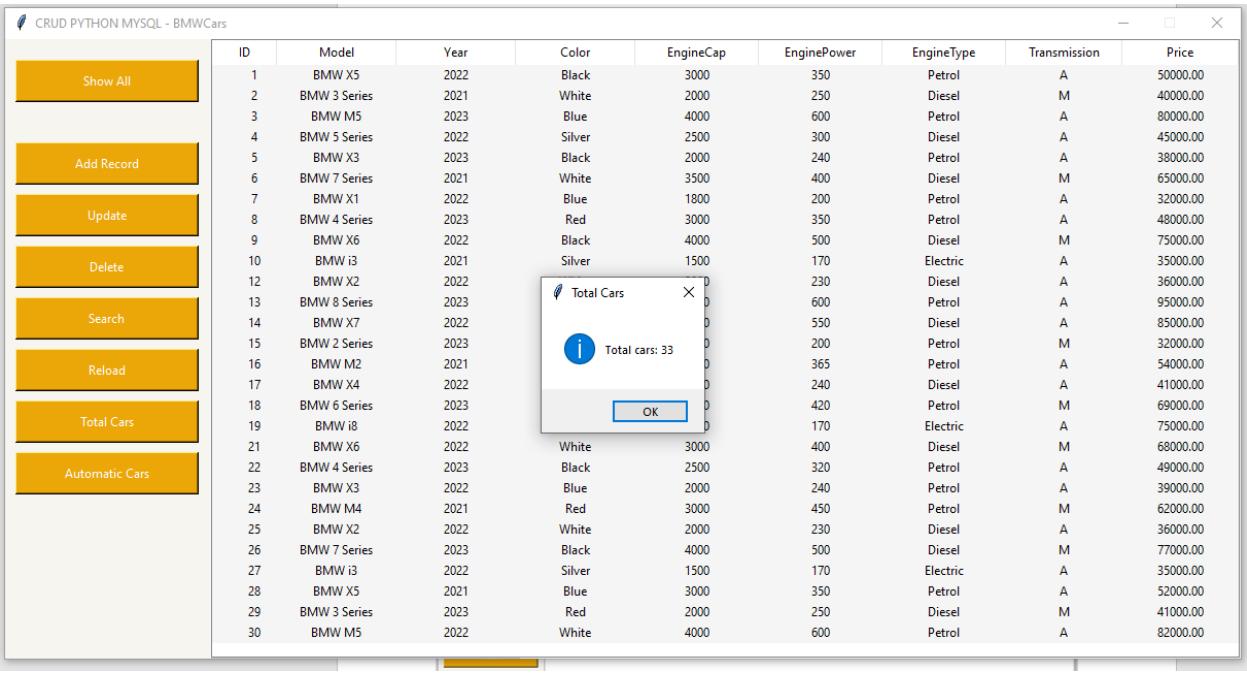
10	BMW i3	2021	Silver	1500	170	Electric	A	35000.00
11	BMW M4	2023	Red	3000	450	Petrol	M	62000.00
12	BMW X2	2022	White	2000	230	Diesel	A	36000.00

SEARCH



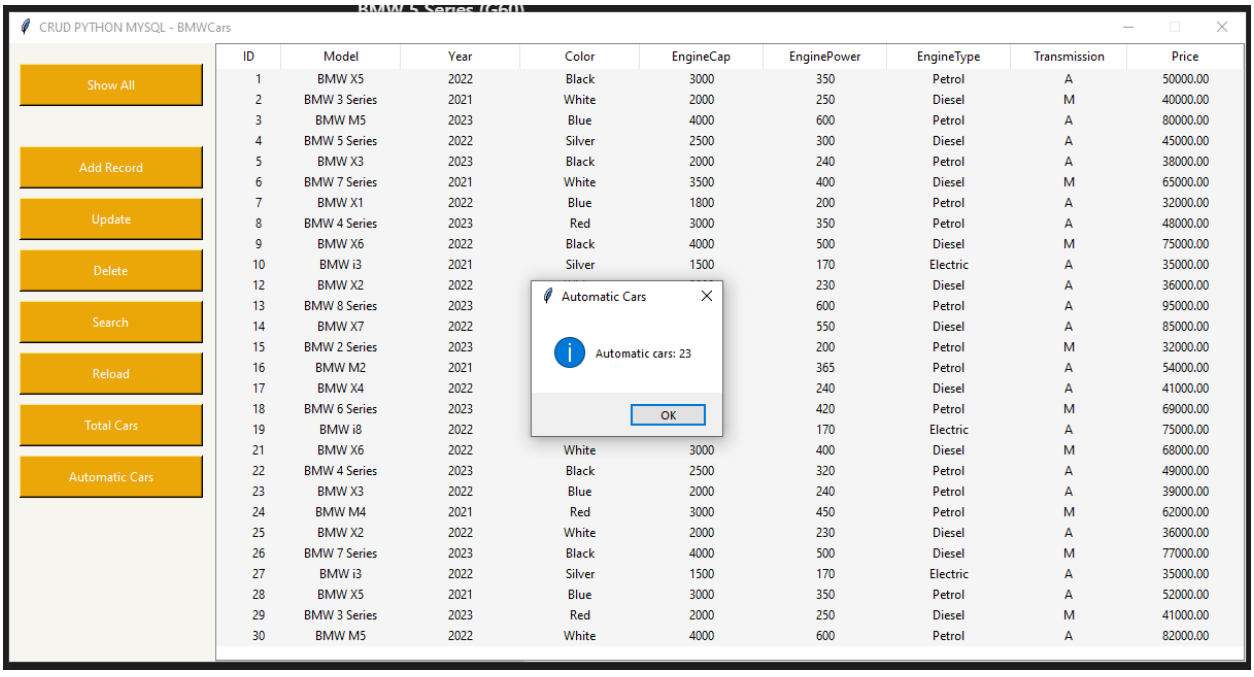
TAYTING, JUSTINE JAY D.
BSCS – C204

TOTAL CARS



ID	Model	Year	Color	EngineCap	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
12	BMW X2	2022			230	Diesel	A	36000.00
13	BMW 8 Series	2023			600	Petrol	A	95000.00
14	BMW X7	2022			550	Diesel	A	85000.00
15	BMW 2 Series	2023			200	Petrol	M	32000.00
16	BMW M2	2021			365	Petrol	A	54000.00
17	BMW X4	2022			240	Diesel	A	41000.00
18	BMW 6 Series	2023			420	Petrol	M	69000.00
19	BMW i8	2022			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
30	BMW M5	2022	White	4000	600	Petrol	A	82000.00

TOTAL AUTOMATIC CARS



ID	Model	Year	Color	EngineCap	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
12	BMW X2	2022			230	Diesel	A	36000.00
13	BMW 8 Series	2023			600	Petrol	A	95000.00
14	BMW X7	2022			550	Diesel	A	85000.00
15	BMW 2 Series	2023			200	Petrol	M	32000.00
16	BMW M2	2021			365	Petrol	A	54000.00
17	BMW X4	2022			240	Diesel	A	41000.00
18	BMW 6 Series	2023			420	Petrol	M	69000.00
19	BMW i8	2022			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
30	BMW M5	2022	White	4000	600	Petrol	A	82000.00