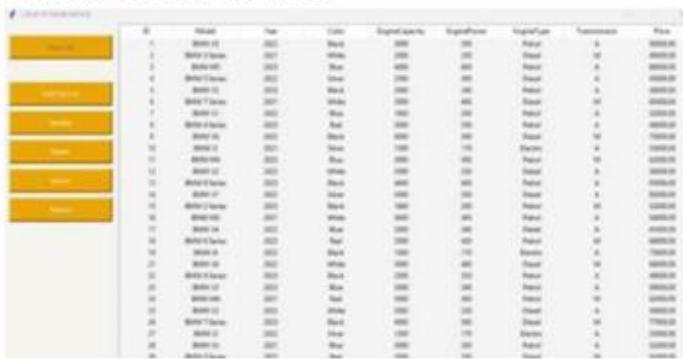


Finals Lab Task 6.
MySQL CRUD Operations In Python Using GUI Tkinter

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(lcs204) 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



ID	Model	Year	Color	OriginalSalePrice	OriginalPurchase	ImportType	Transmission	Price
1	BMW X5	2021	Black	5000	3000	Asian	A	70000
2	BMW X6	2021	White	6000	4000	Asian	A	80000
3	BMW 320i	2021	Blue	4500	3500	Asian	A	60000
4	BMW 7 Series	2021	Grey	10000	8000	Asian	A	120000
5	BMW 520i	2021	White	5000	4000	Asian	A	75000
6	BMW 7 Series	2021	White	10000	8000	Asian	A	120000
7	BMW 320i	2021	White	4500	3500	Asian	A	60000
8	BMW 520i	2021	Blue	5000	4000	Asian	A	75000
9	BMW X3	2021	Black	5000	3000	Asian	A	70000
10	BMW X3	2021	Black	5000	3000	Asian	A	70000
11	BMW X5	2021	Black	5000	3000	Asian	A	70000
12	BMW X5	2021	Black	5000	3000	Asian	A	70000
13	BMW 320i	2021	White	4500	3500	Asian	A	60000
14	BMW 320i	2021	White	4500	3500	Asian	A	60000
15	BMW 320i	2021	White	4500	3500	Asian	A	60000
16	BMW 320i	2021	White	4500	3500	Asian	A	60000
17	BMW 320i	2021	White	4500	3500	Asian	A	60000
18	BMW 320i	2021	White	4500	3500	Asian	A	60000
19	BMW 320i	2021	White	4500	3500	Asian	A	60000
20	BMW 320i	2021	White	4500	3500	Asian	A	60000
21	BMW 320i	2021	White	4500	3500	Asian	A	60000
22	BMW 320i	2021	White	4500	3500	Asian	A	60000
23	BMW 320i	2021	White	4500	3500	Asian	A	60000
24	BMW 320i	2021	White	4500	3500	Asian	A	60000
25	BMW 320i	2021	White	4500	3500	Asian	A	60000
26	BMW 320i	2021	White	4500	3500	Asian	A	60000
27	BMW 320i	2021	White	4500	3500	Asian	A	60000
28	BMW 320i	2021	White	4500	3500	Asian	A	60000
29	BMW 320i	2021	White	4500	3500	Asian	A	60000
30	BMW 320i	2021	White	4500	3500	Asian	A	60000
31	BMW 320i	2021	White	4500	3500	Asian	A	60000
32	BMW 320i	2021	White	4500	3500	Asian	A	60000
33	BMW 320i	2021	White	4500	3500	Asian	A	60000
34	BMW 320i	2021	White	4500	3500	Asian	A	60000
35	BMW 320i	2021	White	4500	3500	Asian	A	60000
36	BMW 320i	2021	White	4500	3500	Asian	A	60000
37	BMW 320i	2021	White	4500	3500	Asian	A	60000
38	BMW 320i	2021	White	4500	3500	Asian	A	60000
39	BMW 320i	2021	White	4500	3500	Asian	A	60000
40	BMW 320i	2021	White	4500	3500	Asian	A	60000
41	BMW 320i	2021	White	4500	3500	Asian	A	60000
42	BMW 320i	2021	White	4500	3500	Asian	A	60000
43	BMW 320i	2021	White	4500	3500	Asian	A	60000
44	BMW 320i	2021	White	4500	3500	Asian	A	60000
45	BMW 320i	2021	White	4500	3500	Asian	A	60000
46	BMW 320i	2021	White	4500	3500	Asian	A	60000
47	BMW 320i	2021	White	4500	3500	Asian	A	60000
48	BMW 320i	2021	White	4500	3500	Asian	A	60000
49	BMW 320i	2021	White	4500	3500	Asian	A	60000
50	BMW 320i	2021	White	4500	3500	Asian	A	60000
51	BMW 320i	2021	White	4500	3500	Asian	A	60000
52	BMW 320i	2021	White	4500	3500	Asian	A	60000
53	BMW 320i	2021	White	4500	3500	Asian	A	60000
54	BMW 320i	2021	White	4500	3500	Asian	A	60000
55	BMW 320i	2021	White	4500	3500	Asian	A	60000
56	BMW 320i	2021	White	4500	3500	Asian	A	60000
57	BMW 320i	2021	White	4500	3500	Asian	A	60000
58	BMW 320i	2021	White	4500	3500	Asian	A	60000
59	BMW 320i	2021	White	4500	3500	Asian	A	60000
60	BMW 320i	2021	White	4500	3500	Asian	A	60000
61	BMW 320i	2021	White	4500	3500	Asian	A	60000
62	BMW 320i	2021	White	4500	3500	Asian	A	60000
63	BMW 320i	2021	White	4500	3500	Asian	A	60000
64	BMW 320i	2021	White	4500	3500	Asian	A	60000
65	BMW 320i	2021	White	4500	3500	Asian	A	60000
66	BMW 320i	2021	White	4500	3500	Asian	A	60000
67	BMW 320i	2021	White	4500	3500	Asian	A	60000
68	BMW 320i	2021	White	4500	3500	Asian	A	60000
69	BMW 320i	2021	White	4500	3500	Asian	A	60000
70	BMW 320i	2021	White	4500	3500	Asian	A	60000
71	BMW 320i	2021	White	4500	3500	Asian	A	60000
72	BMW 320i	2021	White	4500	3500	Asian	A	60000
73	BMW 320i	2021	White	4500	3500	Asian	A	60000
74	BMW 320i	2021	White	4500	3500	Asian	A	60000
75	BMW 320i	2021	White	4500	3500	Asian	A	60000
76	BMW 320i	2021	White	4500	3500	Asian	A	60000
77	BMW 320i	2021	White	4500	3500	Asian	A	60000
78	BMW 320i	2021	White	4500	3500	Asian	A	60000
79	BMW 320i	2021	White	4500	3500	Asian	A	60000
80	BMW 320i	2021	White	4500	3500	Asian	A	60000
81	BMW 320i	2021	White	4500	3500	Asian	A	60000
82	BMW 320i	2021	White	4500	3500	Asian	A	60000
83	BMW 320i	2021	White	4500	3500	Asian	A	60000
84	BMW 320i	2021	White	4500	3500	Asian	A	60000
85	BMW 320i	2021	White	4500	3500	Asian	A	60000
86	BMW 320i	2021	White	4500	3500	Asian	A	60000
87	BMW 320i	2021	White	4500	3500	Asian	A	60000
88	BMW 320i	2021	White	4500	3500	Asian	A	60000
89	BMW 320i	2021	White	4500	3500	Asian	A	60000
90	BMW 320i	2021	White	4500	3500	Asian	A	60000
91	BMW 320i	2021	White	4500	3500	Asian	A	60000
92	BMW 320i	2021	White	4500	3500	Asian	A	60000
93	BMW 320i	2021	White	4500	3500	Asian	A	60000
94	BMW 320i	2021	White	4500	3500	Asian	A	60000
95	BMW 320i	2021	White	4500	3500	Asian	A	60000
96	BMW 320i	2021	White	4500	3500	Asian	A	60000
97	BMW 320i	2021	White	4500	3500	Asian	A	60000
98	BMW 320i	2021	White	4500	3500	Asian	A	60000
99	BMW 320i	2021	White	4500	3500	Asian	A	60000
100	BMW 320i	2021	White	4500	3500	Asian	A	60000

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/1e6Eh55qLAweptf0A_I8GKh70eM6jAxJ?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

Source Code:

(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}, {capacity}, {power},
                           {type}, {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}', capacity='{capacity}', power='{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 []

```

(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="cs204", 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 - BMNCars")
        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, background="#e6a607",
                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="#e6a607",
foreground="white")
self.buttonNew.place(x=10, y=100)

self.buttonUpdate = tk.Button(frame1, text="Update",
command=self.UpdateData,
width=24, height=2,
background="#e6a607", foreground="white")
self.buttonUpdate.place(x=10, y=150)

self.buttonDelete = tk.Button(frame1, text="Delete",
command=self.DeleteData,
width=24, height=2,
background="#e6a607", foreground="white")
self.buttonDelete.place(x=10, y=200)

self.buttonSearch = tk.Button(frame1, text="Search",
command=self.SearchData,
width=24, height=2,
background="#e6a607", foreground="white")
self.buttonSearch.place(x=10, y=250)

self.buttonReload = tk.Button(frame1, text="Reload",
command=self.fnInit,
width=24, height=2,
background="#e6a607", foreground="white")
self.buttonReload.place(x=10, y=300)

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

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)

        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")

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)

def show_info(self):
    self.cnn.connect()
    data = self.cnn.execute select("car")
    total records = len(data)
    highest price car = max(data, key=lambda x: x[8]) if data else None
    highest price model = highest price car[1] if highest price car else
    "N/A"
    manual count = sum(1 for row in data if row[7].upper() == "M")
    automatic count = sum(1 for row in data if row[7].upper() == "A")
    self.cnn.disconnect()

    messagebox.showinfo("Car Info",
                        f"Total Records: {total records}\n"
                        f"Car with Highest Price:\n"
                        f"{highest price model}\n"
                        f"Total Manual Cars: {manual count}\n"
                        f"Total Automatic Cars: {automatic count}")

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.buttonReload]:
            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.buttonReload]:
            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.buttonReload):
            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.buttonReload):
                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.entry1, self.entry3, self.entry4, self.entry5, self.entry6, self.entry7,
                               self.entry8, self.entry9]):
                        entry.insert(0, values[i])
                self.entry3.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.cnx.connect()
            self.cnx.execute delete("car", id_selectioned)
            self.cnx.disconnect()
            self.f1init()

    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((widthWindow+heightWindow+pwidth)+(pheight
        - 60))

    def show_search_data(l, 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{3, 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)

```

(main.py)

```

import tkinter as tk
import window

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

if __name__ == "__main__":
    main()

```

Sample Output:

Show All:

CRUD PYTHON MYSQL - BMWCars								
ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
1	BMW X1	2022	Blue	1800	200	Petrol	A	32000.00
2	BMW 4 Series	2023	Red	2000	220	Petrol	A	48000.00
3	BMW X5	2022	Black	4000	300	Diesel	M	70000.00
4	BMW i3	2021	Grey	1500	170	Electric	A	35000.00
5	BMW M3	2023	Blue	3000	320	Petrol	M	62000.00
6	BMW X2	2022	White	2000	230	Diesel	A	38000.00
7	BMW 3 Series	2023	Black	4000	300	Petrol	A	95000.00
8	BMW X7	2022	Silver	4500	350	Diesel	A	135000.00
9	BMW Z4	2023	Black	1800	200	Petrol	M	33000.00
10	BMW M2	2021	White	2000	280	Petrol	A	54000.00
11	BMW M4	2022	Blue	3000	340	Diesel	A	47000.00
12	BMW 5 Series	2023	Red	2500	220	Petrol	M	69000.00
13	BMW 6 Series	2022	Black	3000	250	Diesel	A	79000.00
14	BMW X6	2022	Silver	4000	350	Diesel	A	125000.00
15	BMW 2 Series	2023	Black	2000	200	Petrol	M	30000.00
16	BMW M5	2021	White	3000	380	Petrol	A	140000.00
17	BMW M6	2022	Blue	3500	400	Petrol	A	150000.00
18	BMW 7 Series	2023	Red	4000	380	Diesel	M	170000.00
19	BMW 8 Series	2022	Black	3500	320	Diesel	A	180000.00
20	BMW X3	2023	White	2500	220	Diesel	M	55000.00
21	BMW 4 Series	2023	Black	2500	230	Petrol	A	49000.00
22	BMW X2	2022	Blue	2000	240	Petrol	M	39000.00
23	BMW M5	2021	Red	3000	300	Petrol	M	63000.00
24	BMW X2	2022	White	2000	230	Diesel	A	36000.00
25	BMW 7 Series	2023	Black	4000	350	Diesel	M	175000.00
26	BMW i3	2022	Grey	1500	170	Electric	A	31000.00
27	BMW 3 Series	2023	Blue	3000	300	Petrol	A	32000.00
28	BMW Z5	2021	White	2000	200	Petrol	A	41000.00
29	BMW 2 Series	2023	Red	2500	220	Diesel	M	41000.00
30	BMW M6	2022	White	4000	380	Petrol	A	120000.00
31	BMW X1	2023	Black	1800	200	Petrol	A	32000.00
32	BMW 3 Series	2021	Silver	2000	230	Diesel	A	47000.00
33	BMW X7	2022	Black	4000	350	Diesel	A	170000.00
34	BMW 2 Series	2023	Blue	2000	200	Petrol	M	34000.00
35	BMW M2	2022	Red	2500	280	Petrol	A	110000.00

Add:

CRUD PYTHON MYSQL - BMWCars								
ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
20	BMW 3E-40	2021	Black	4000	300	Diesel	M	75000.00
Model	BMW 3E-40	2021	Black	4000	300	Diesel	M	75000.00
Year-Make	1967	2021	Black	4000	300	Diesel	M	75000.00
Color	White	2021	Black	4000	300	Diesel	M	75000.00
Engine Capacity	1500	2021	Black	4000	300	Diesel	M	75000.00
Brand	BMW	2021	Black	4000	300	Diesel	M	75000.00
Engine Power	240	2021	Black	4000	300	Diesel	M	75000.00
Engine Type	petrol	2021	Black	4000	300	Diesel	M	75000.00
Transmission Type	M	2021	Black	4000	300	Diesel	M	75000.00
Price	65000	2021	Black	4000	300	Diesel	M	75000.00

ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
18	BMW 6 Series	2023	Red	3500	420	Petrol	M	69000.00
19	BMW i8	2022	Black	1500	170	Electric	A	75000.00
20	BMW 3E46	1997	White	1500	240	petrol	M	85000.00
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

Update:

CRUD PYTHON MySQL - BMW Cars

The screenshot shows a Python application window titled "CRUD PYTHON MySQL - BMW Cars". It contains two main sections: a form on the left and a table on the right.

Form (Left):

- Model: BMW X5
- Year Make: 2022
- Color: Blue
- Engine Capacity: 2000
- Engine Power: 350
- Engine Type: Petrol
- Transmission Type: A
- Price: 60000.00

Table (Right):

ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
1	BMW X2	2022	Black	2000	200	Petrol	A	40000.00
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 1 Series	2022	Silver	2500	300	Diesel	A	45000.00
5	BMW X3	2023	Black	2500	320	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	2000	230	Petrol	A	48000.00
9	BMW M6	2022	Black	4000	500	Diesel	M	75000.00
10	BMW Z4	2021	Silver	1500	170	Electric	A	55000.00
11	BMW M4	2023	Blue	3000	400	Petrol	M	62000.00
12	BMW i2	2022	White	2000	230	Diesel	A	36000.00
13	BMW 5 Series	2023	Black	4000	600	Petrol	A	90000.00
14	BMW X7	2022	Silver	4500	350	Diesel	A	85000.00
15	BMW 2 Series	2023	Black	1800	200	Petrol	M	32000.00
16	BMW M2	2021	White	3000	380	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	68000.00
19	BMW M	2022	Black	1800	170	Electric	A	78000.00
20	BMW 540i	1997	White	1500	240	petrol	M	55000.00
21	BMW M8	2022	White	3500	400	Diesel	M	68000.00
22	BMW 4 Series	2023	Black	2500	320	Petrol	A	49000.00
23	BMW i3	2022	Blue	2000	240	Petrol	A	38000.00
24	BMW M4	2021	Red	3000	400	Petrol	M	62000.00
25	BMW i2	2022	White	2000	230	Diesel	A	34000.00
26	BMW 7 Series	2023	Black	4500	500	Diesel	M	77000.00
27	BMW i6	2022	Silver	1800	170	Electric	A	58000.00
28	BMW X5	2021	Blue	3000	350	Petrol	A	52000.00

ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
1	BMW X5	2022	Red	2000	350	Petrol	A	60000.00
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

Search:

localhost:51000 - search.cshtml

ID	Model	Year	Color	EngineCapacity	EnginePower	EngineType	Transmission	Price
1	BMW X1	2022	Red	2000	150	Petrol	A	60000.00
2	BMW 3 Series	2021	White	2000	200	Diesel	M	40000.00
3	BMW M5	2022	Blue	4000	400	Petrol	A	90000.00
4	BMW 5 Series	2022	Silver	2500	180	Diesel	A	45000.00
5	BMW X3	2022	Black	2000	240	Petrol	A	38000.00
6	BMW 7 Series	2021	White	2500	200	Diesel	M	65000.00
7	BMW X1	2022	Blue	1800	160	Petrol	A	32000.00
8	BMW 4 Series	2022	Red	2000	180	Petrol	A	48000.00
9	BMW X6	2022	Black	4000	350	Diesel	M	75000.00
10	BMW i3	2021	Silver	1500	170	Electric	A	30000.00
11	BMW 540e	2022	Blue	3000	200	Petrol	M	62000.00
12	BMW iX	2022	White	2500	220	Diesel	A	36000.00
13	BMW 8 Series	2022	Black	4000	400	Petrol	A	95000.00
14	BMW i4	2022	Black	1500	170	Electric	A	35000.00
15	BMW X6	2022	White	3000	200	Diesel	M	68000.00
16	BMW 4 Series	2022	Black	2000	180	Petrol	A	49000.00
17	BMW X3	2022	Blue	2000	240	Petrol	A	39000.00
18	BMW 3 Series	2021	Red	2000	180	Diesel	M	42000.00
19	BMW X2	2022	White	2000	180	Diesel	M	37000.00
20	BMW 2 Series	2022	Red	2000	180	Petrol	A	34000.00
21	BMW X1	2022	Blue	2000	180	Petrol	A	33000.00
22	BMW 5 Series	2022	Black	2500	180	Diesel	M	45000.00
23	BMW X3	2022	Blue	2000	240	Petrol	A	38000.00
24	BMW 3 Series	2021	Red	2000	180	Petrol	M	43000.00
25	BMW X2	2022	White	2000	180	Diesel	A	36000.00
26	BMW 7 Series	2021	Black	4000	350	Diesel	M	75000.00
27	BMW i3	2021	Silver	1500	170	Electric	A	30000.00
28	BMW X3	2021	Blue	2000	180	Petrol	A	32000.00
29	BMW 5 Series	2021	Red	2000	200	Diesel	M	40000.00

Search:

Model:

Year Model:

Color:

Advanced Options: Search ID Model Year Price

Engine Type:

Transmission Type:

Price:

Show Info:

