Python SQL CS project

Topic : Sushant Computer Builder

Name – Sushant Sejwal

Class – 12th A

Teacher – PawanJeet Singh

Certificate

Name : Sushant Sejwal

Class : 12th A Roll No : 39

School : Kendriya Vidyalaya No. 2 Delhi Cantt – 110010

This is certified to be the bonafide work of the student in the Python File Handling Project (binary file) during the year 2022-2023

Teacher : PawanJeet Singh

External Examiner

Acknowledgement

I, Sushant Sejwal of class 12th A will express my sincere gratitude to my computer teacher Mr. Pawanjeet Singh for their support, guidance, teaching and encouragement for the 2 year time period from the starting class 11th to the ending of class 12th.

Index

Sushant PC Builder

Overview of Project

This project is a replica of a custom PC builder website or shop where people can build custom computer for them or they can buy individuals parts for their computer.

In order to run the program you first need data in SQL tables. The tables are used to store data of computer parts that are needed by the program to run the program.

To download the file that contain all the SQL code in order create set up database for the rest of the program use this link: https://github.com/SushantSejwal/SQL-Project

There are total 9 files and each file contains functions which are used to perform task like authentication, showing products, buying individual parts, show products in cart, remove products from the cart, show bill, get informations about product and one main file to run the whole program.

There are 8 command which the person has to to use while running the program.

Let's see all of the commands one by one.

1. **help** : this command will show all the commands that user can use .
2. **show** : this command will show all the products that present

in the database.

3. **custom** : this command will be used to build the custom computers.
4. **buy** : this command will be used to buy individual computer parts from the store.
5. **cart** : this command will show all the items that are present in the cart.
6. **rm** : this command will be used to remove products from the cart.
7. **bill** : this command will show the total price of the items that are present in the cart.
8. **exit** : this command will be used to terminate the program.

System Requirement

Hardware : any machine, computer, laptop, phone that can run linux (Arch, Debian, Red Hat, Open Suse, etc) or windows or macOS to run the program is OK.

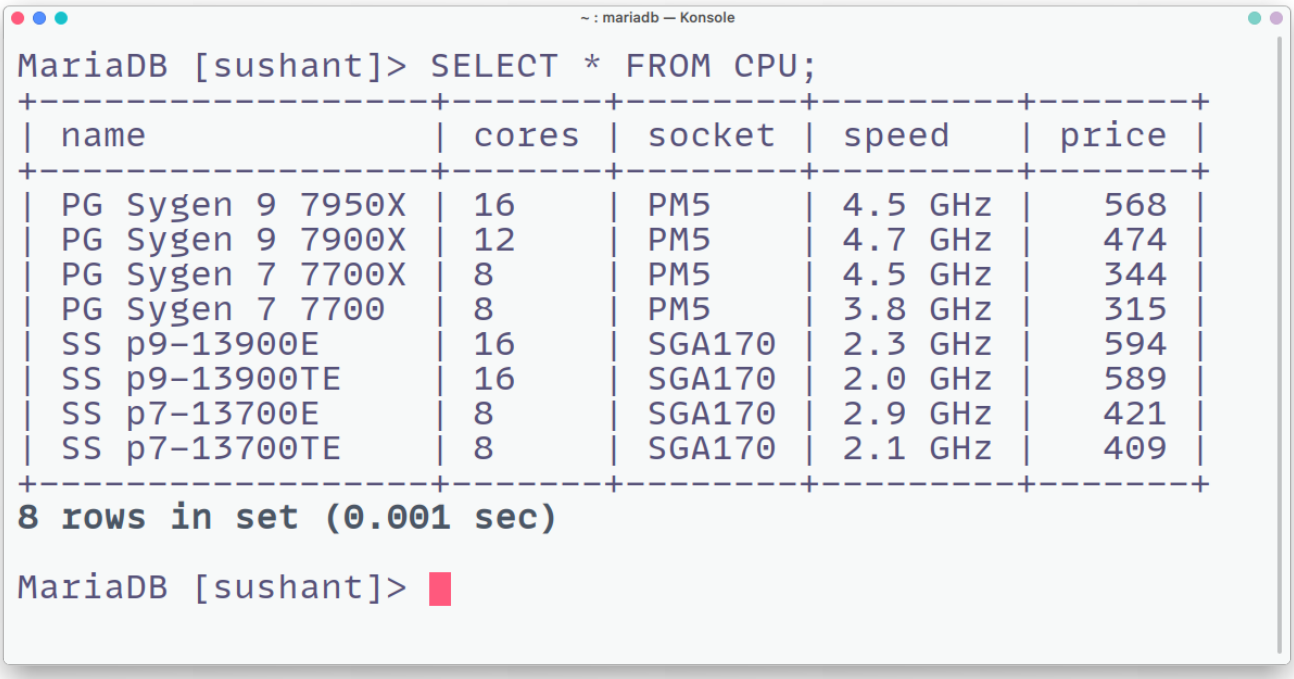
Software : any Operating system is OK that can run python and MySQL like, linux (Arch, Debian, Red Hat, Open Suse, etc) or windows or macOS.

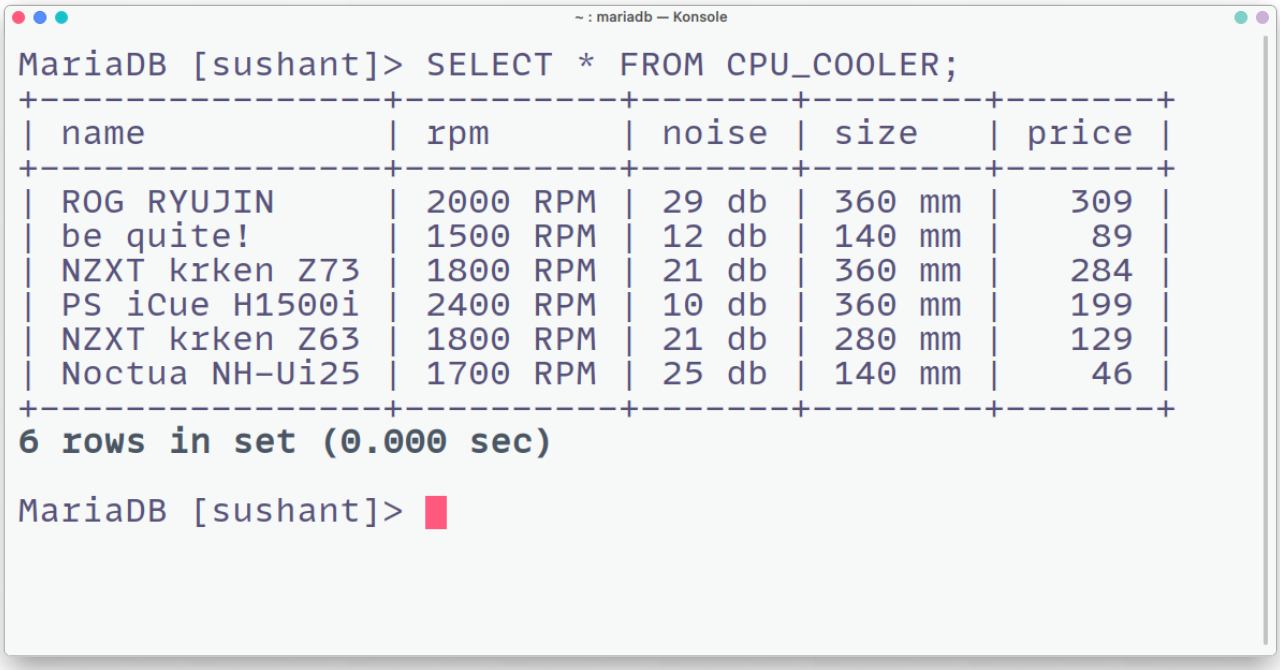
Python package **mysql-connector** should be installed with python.

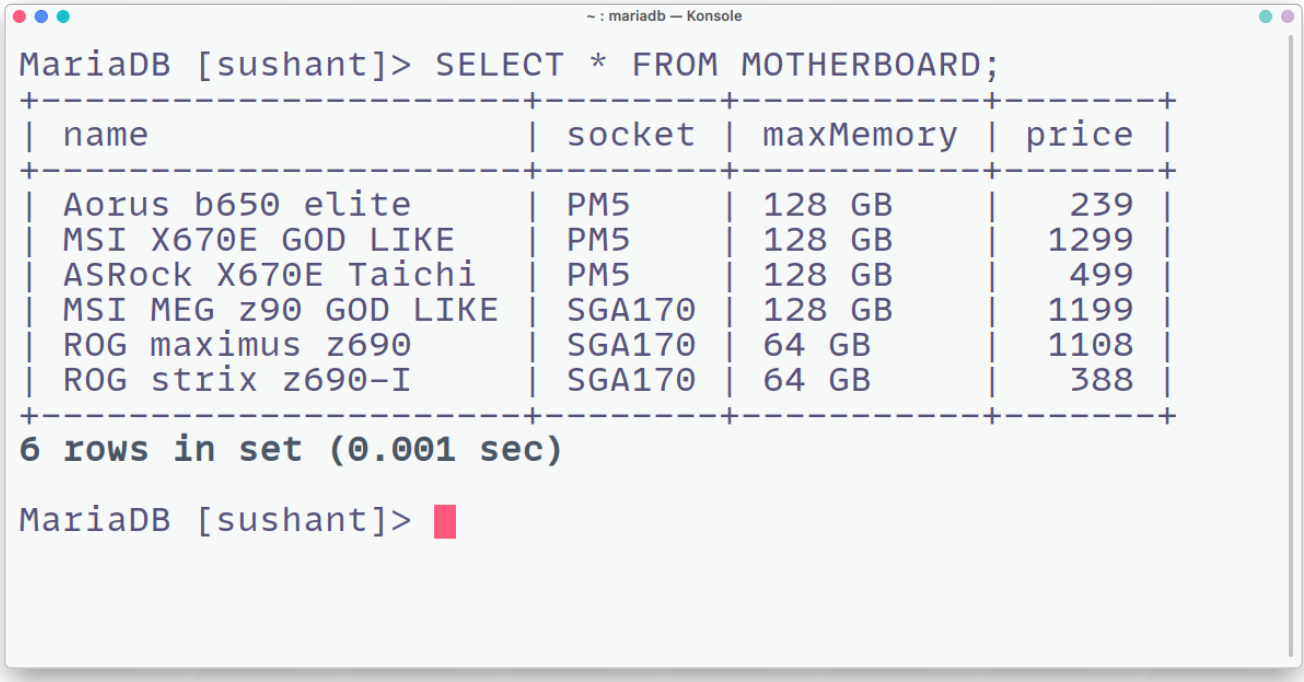
You use this command to install **mysql-connector**

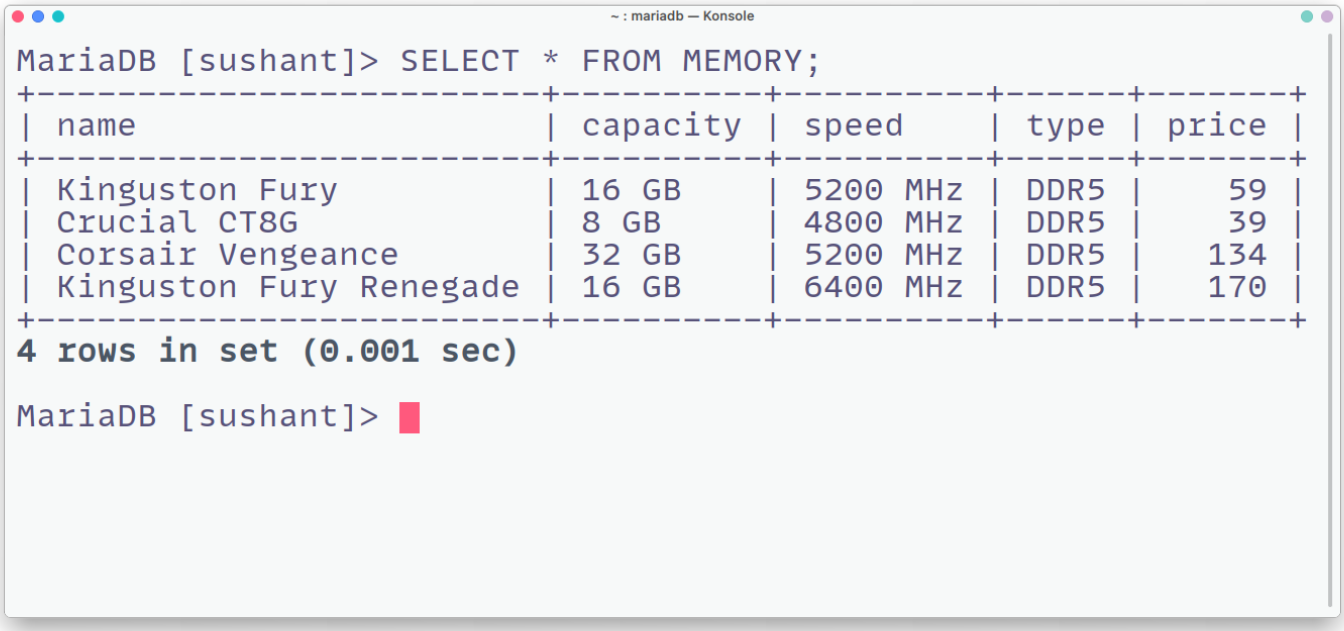
```
pip install mysql-connector
```

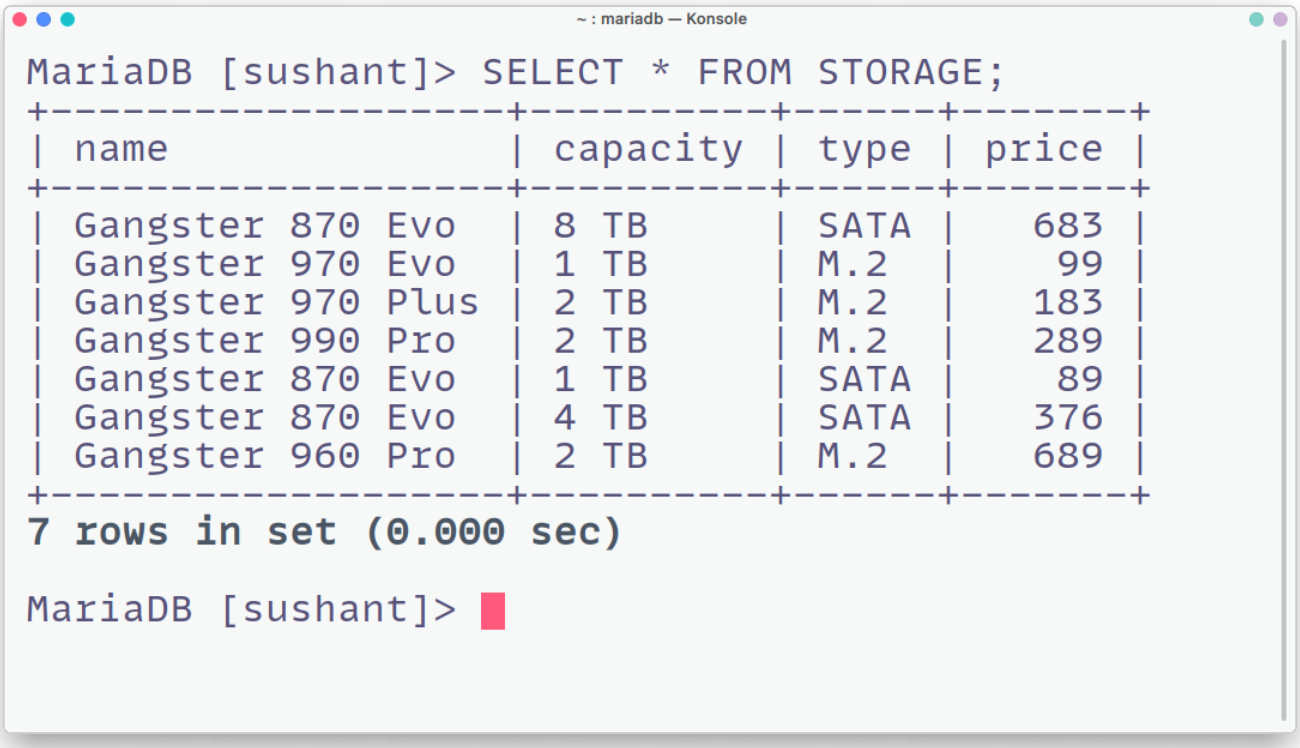
ScreenShots of Data Required

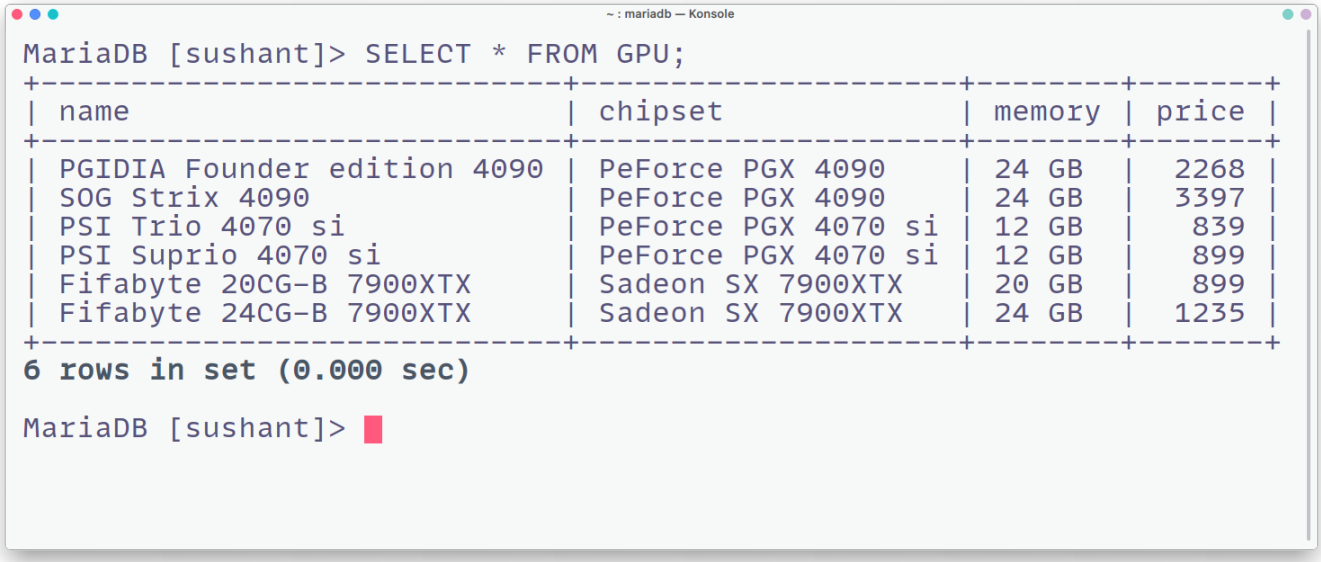





~ : mariadb — Konsole
MariaDB [sushant]> SELECT * FROM MOTHERBOARD;
name	socket	maxMemory	price
Aorus b650 elite	PM5	128 GB	239
MSI X670E GOD LIKE	PM5	128 GB	1299
ASRock X670E Taichi	PM5	128 GB	499
MSI MEG z90 GOD LIKE	SGA170	128 GB	1199
ROG maximus z690	SGA170	64 GB	1108
ROG strix z690-I	SGA170	64 GB	388
6 rows in set (0.001 sec)
MariaDB [sushant]>


~ : mariadb — Konsole
MariaDB [sushant]> SELECT * FROM MEMORY;
name	capacity	speed	type	price
Kinguston Fury	16 GB	5200 MHz	DDR5	59
Crucial CT8G	8 GB	4800 MHz	DDR5	39
Corsair Vengeance	32 GB	5200 MHz	DDR5	134
Kinguston Fury Renegade	16 GB	6400 MHz	DDR5	170
4 rows in set (0.001 sec)
MariaDB [sushant]>


~ : mariadb — Konsole
MariaDB [sushant]> SELECT * FROM STORAGE;
name	capacity	type	price
Gangster 870 Evo	8 TB	SATA	683
Gangster 970 Evo	1 TB	M.2	99
Gangster 970 Plus	2 TB	M.2	183
Gangster 990 Pro	2 TB	M.2	289
Gangster 870 Evo	1 TB	SATA	89
Gangster 870 Evo	4 TB	SATA	376
Gangster 960 Pro	2 TB	M.2	689
7 rows in set (0.000 sec)
MariaDB [sushant]>


~ : mariadb — Konsole
MariaDB [sushant]> SELECT * FROM GPU;
name	chipset	memory	price
PGIDIA Founder edition 4090	PeForce PGX 4090	24 GB	2268
SOG Strix 4090	PeForce PGX 4090	24 GB	3397
PSI Trio 4070 si	PeForce PGX 4070 si	12 GB	839
PSI Suprio 4070 si	PeForce PGX 4070 si	12 GB	899
Fifabyte 20CG-B 7900XTX	Sadeon SX 7900XTX	20 GB	899
Fifabyte 24CG-B 7900XTX	Sadeon SX 7900XTX	24 GB	1235
6 rows in set (0.000 sec)
MariaDB [sushant]>

~ : mariadb — Konsole

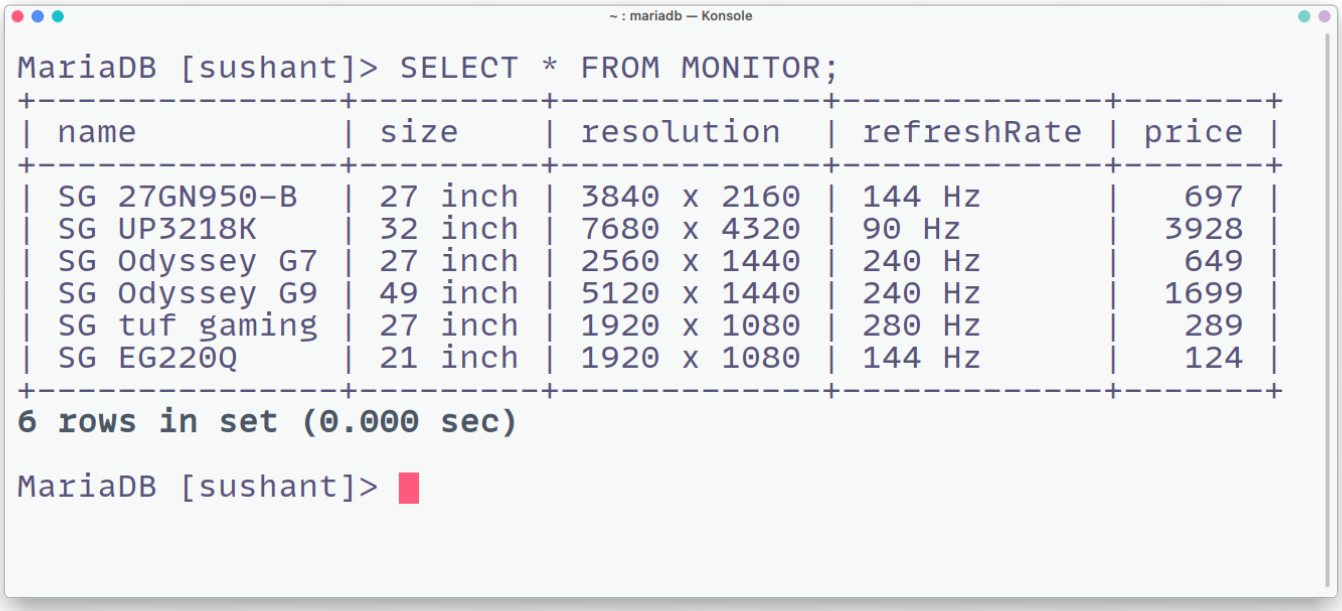
```
MariaDB [sushant]> SELECT * FROM PSU;
+----------------+------+--------------+-----------+-------+
| name           | type | efficiency   | wattage   | price |
+----------------+------+--------------+-----------+-------+
| pegsus RM850X  | ATX  | 80+ Gold     | 850 watt  |   137 |
| pegsus RM1000X | ATX  | 80+ Gold     | 1000 watt |   189 |
| pegsus HX1200  | ATX  | 80+ Platinum | 1200 watt |   289 |
| pegsus AX1000i | ATX  | 80+ Titanium | 1600 watt |   609 |
| pegsus Awesome | ATX  | 80+ Platinum | 1300 watt |   481 |
+----------------+------+--------------+-----------+-------+
5 rows in set (0.001 sec)

MariaDB [sushant]>
```

~ : mariadb — Konsole

```
MariaDB [sushant]> SELECT * FROM CABINET;
+-----------------+------+-------+
| name            | type | price |
+-----------------+------+-------+
| Corsair 4000D   | ATX  |   104 |
| Corsair iCUE    | ATX  |   336 |
| ASUS ROG helios | EATX |   319 |
| Corsair 5000X   | EATX |   214 |
+-----------------+------+-------+
4 rows in set (0.000 sec)

MariaDB [sushant]>
```



Source Code

file : sql-code.sql (include all the code to set up database)

```sql
CREATE DATABASE sushant;

USE sushant;

CREATE TABLE CPU (
    name VARCHAR(35),
    cores VARCHAR(10),
    socket VARCHAR(10),
    speed VARCHAR(10),
    price INT
);

CREATE TABLE CPU_COOLER(
    name VARCHAR(35),
    rpm VARCHAR(10),
    noise VARCHAR(10),
    size VARCHAR(10),
    price INT
);
```

```sql
CREATE TABLE MOTHERBOARD (
    name VARCHAR(35),
    socket VARCHAR(10),
    maxMemory VARCHAR(10),
    price INT
);

CREATE TABLE MEMORY (
    name VARCHAR(35),
    capacity VARCHAR(10),
    speed VARCHAR(10),
    type VARCHAR(10),
    price INT
);

CREATE TABLE STORAGE (
    name VARCHAR(35),
    capacity VARCHAR(510),
    type VARCHAR(10),
    price INT
);

CREATE TABLE GPU (
    name VARCHAR(35),
    chipset VARCHAR(30),
    memory VARCHAR(10),
    price INT
);

CREATE TABLE CABINET (
    name VARCHAR(35),
    type VARCHAR(10),
    price INT
);

CREATE TABLE PSU (
    name VARCHAR(35),
    type VARCHAR(10),
    efficiency VARCHAR(20),
    wattage VARCHAR(10),
    price INT
);

CREATE TABLE MONITOR (
    name VARCHAR(35),
    size VARCHAR(10),
    resolution VARCHAR(20),
```

```
    refreshRate VARCHAR(10),
    price INT
);

INSERT INTO CPU VALUES
    ("PG Sygen 9 7950X", "16", "PM5", "4.5 GHz", 568),
    ("PG Sygen 9 7900X", "12", "PM5", "4.7 GHz", 474),
    ("PG Sygen 7 7700X", "8", "PM5", "4.5 GHz", 344),
    ("PG Sygen 7 7700", "8", "PM5", "3.8 GHz", 315);

INSERT INTO CPU VALUES
    ("SS p9-13900E", "16", "SGA170", "2.3 GHz", 594),
    ("SS p9-13900TE", "16", "SGA170", "2.0 GHz", 589),
    ("SS p7-13700E", "8", "SGA170", "2.9 GHz", 421),
    ("SS p7-13700TE", "8", "SGA170", "2.1 GHz", 409);


INSERT INTO CPU_COOLER VALUES
    ("ROG RYUJIN", "2000 RPM", "29 db", "360 mm", 309),
    ("be quite!", "1500 RPM", "12 db", "140 mm", 89),
    ("NZXT krken Z73", "1800 RPM", "21 db", "360 mm", 284),
    ("PS iCue H1500i", "2400 RPM", "10 db", "360 mm", 199),
    ("NZXT krken Z63", "1800 RPM", "21 db", "280 mm", 129),
    ("Noctua NH-Ui25", "1700 RPM", "25 db", "140 mm", 46);

INSERT INTO MOTHERBOARD VALUES
    ("Aorus b650 elite", "PM5", "128 GB", 239),
    ("MSI X670E GOD LIKE", "PM5", "128 GB", 1299),
    ("ASRock X670E Taichi", "PM5", "128 GB", 499),
    ("MSI MEG z90 GOD LIKE", "SGA170", "128 GB", 1199),
    ("ROG maximus z690", "SGA170", "64 GB", 1108),
    ("ROG strix z690-I", "SGA170", "64 GB", 388);

INSERT INTO MEMORY VALUES
    ("Kinguston Fury", "16 GB", "5200 MHz", "DDR5", 59),
    ("Crucial CT8G", "8 GB", "4800 MHz", "DDR5", 39),
    ("Corsair Vengeance", "32 GB", "5200 MHz", "DDR5", 134),
    ("Kinguston Fury Renegade", "16 GB", "6400 MHz", "DDR5", 170);

INSERT INTO STORAGE VALUES
    ("Gangster 870 Evo", "8 TB", "SATA", 683),
    ("Gangster 970 Evo", "1 TB", "M.2", 99),
    ("Gangster 970 Plus", "2 TB", "M.2", 183),
    ("Gangster 990 Pro", "2 TB", "M.2", 289),
    ("Gangster 870 Evo", "1 TB", "SATA", 89),
    ("Gangster 870 Evo", "4 TB", "SATA", 376),
    ("Gangster 960 Pro", "2 TB", "M.2", 689);
```

```
INSERT INTO GPU VALUES
    ("PGIDIA Founder edition 4090", "PeForce PGX 4090", "24 GB", 2268),
    ("SOG Strix 4090", "PeForce PGX 4090", "24 GB", 3397),
    ("PSI Trio 4070 si", "PeForce PGX 4070 si", "12 GB", 839),
    ("PSI Suprio 4070 si", "PeForce PGX 4070 si", "12 GB", 899),
    ("Fifabyte 20CG-B 7900XTX", "Sadeon SX 7900XTX", "20 GB", 899),
    ("Fifabyte 24CG-B 7900XTX", "Sadeon SX 7900XTX", "24 GB", 1235);

INSERT INTO CABINET VALUES
    ("Corsair 4000D", "ATX", 104),
    ("Corsair iCUE", "ATX", 336),
    ("ASUS ROG helios", "EATX", 319),
    ("Corsair 5000X", "EATX", 214);

INSERT INTO PSU VALUES
    ("pegsus RM850X", "ATX", "80+ Gold", "850 watt", 137),
    ("pegsus RM1000X", "ATX", "80+ Gold", "1000 watt", 189),
    ("pegsus HX1200", "ATX", "80+ Platinum", "1200 watt", 289),
    ("pegsus AX1000i", "ATX", "80+ Titanium", "1600 watt", 609),
    ("pegsus Awesome", "ATX", "80+ Platinum", "1300 watt", 481);

INSERT INTO MONITOR VALUES
    ("SG 27GN950-B", "27 inch", "3840 x 2160", "144 Hz", 697),
    ("SG UP3218K", "32 inch", "7680 x 4320", "90 Hz", 3928),
    ("SG Odyssey G7", "27 inch", "2560 x 1440", "240 Hz", 649),
    ("SG Odyssey G9", "49 inch", "5120 x 1440", "240 Hz", 1699),
    ("SG tuf gaming", "27 inch", "1920 x 1080", "280 Hz", 289),
    ("SG EG220Q", "21 inch", "1920 x 1080", "144 Hz", 124);
```

file : ss-main.py (main file)

```
# importing time for some delay in output
from time import sleep

# importing modules to run the program
import user
import help
import show
import custom
import buy
import bag
import remove
import bill

# authentication step
userName, password, accoundExist = user.userLogin()
if accoundExist:
```

```
    print("\n\n**-**-**-** WELCOME TO SUSHANT COMPUTER BULDER **-**-
**-**")
    print(" here you can build PCs and buy parts")

    print(f"\n\n **-**-** Welcome {userName} **-**-**\n")

# running program
while accoundExist:
    cmd = input("Enter command or type 'help' for help\n → ")
    cmd = cmd.lower().strip()
    sleep(0.1)

    if cmd == "help" or cmd =="h":
        help.commands()

    elif cmd == "show":
        show.showProduct()

    elif cmd == "custom":
        custom.builder(userName)

    elif cmd == "buy":
        buy.purchase(userName)

    elif cmd == "cart":
        bag.cart(userName)

    elif cmd == "rm":
        remove.remove(userName)

    elif cmd == "bill":
        bill.total(userName)

    elif cmd == "exit" or cmd == "e":
        exit()
    else:
        print("whoopse")
        print()

else:
print("Login is required in order to proceed")
```

file : user.py (account authentication)

```
import mysql.connector as sql
```

```
database = sql.connect(
    host = "localhost",
    user = "root",
    password = ""
)
executer = database.cursor()

# *** checking database exist or not *** #
executer.execute("SHOW databases;")
databaseList = executer.fetchall()
databases = []
for i in databaseList :
    databases.append(i[0])

if "consumer" in databases:
    executer.close()
    database = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    executer = database.cursor()
else:
    executer.execute("CREATE DATABASE consumer;")
    database.commit()
    executer.close()
    database = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    executer = database.cursor()

# *** checking table exist or not *** #
executer.execute("SHOW tables;")
tableList = executer.fetchall()
tables = []
for i in tableList :
    tables.append(i[0])

if "ACCOUNTS" not in tables:
    executer.execute("""
        CREATE TABLE ACCOUNTS (
            username varchar(35) not null PRIMARY KEY,
            password varchar(15) not null
```

```
        );
    """)
    database.commit()

executer.execute("SELECT * FROM ACCOUNTS")
userData = executer.fetchall()

def createAccount():
    while True:
        userName = input("Enter your username : ")
        password = input(f"Enter password for {userName} (max 15 char): ")
        sure = input(f"\n User Name : {userName}\n Password : {password} \
\nare you sure these values are correct ? [yes/NO] default(NO) : ")

        if sure == "y" or sure == "yes":
            executer.execute(f"""
                insert into ACCOUNTS values
                    ("{userName}", "{password}")
                """)
            database.commit()
            print("account created")
            break
        else:
            print("\n\n")
            continue


    return(userName, password, True)



# ==========*** sure ***==========# 
def userLogin():

    while True :
        logIn = input("do you have a account resistered ? [yes/no] : ")
        logIn = logIn.lower().strip()
        accountFound = False
        accountCreated = False

    if logIn == "y" or logIn == "yes":
        userName = input("Enter your username : ")
        password = input(f"Enter password for {userName} : ")
        for data in userData:
            if userName == data[0] and password == data[1]:
                returningName = data[0]
                returningPassword = data[1]
                accountFound = True
```

```
                accountCreated = True
                break
        else:
            print("whopse, account not found\n")
            createChoice = input("Would you like to create new account ?
[yes/no] : ")
            createChoice = createChoice.lower().strip()

            if createChoice == "yes" or createChoice == "y":
                returningName, returningPassword, accountCreated =
createAccount()
                break

            elif createChoice == "no" or createChoice == "n":
                returningName = None
                returningPassword = None
                accountCreated = False
                break

        if accountFound :
            break
    elif logIn == "no" or logIn == "n":
        createChoice = input("Would you like to create new account ?
[yes/no] : ")
        createChoice = createChoice.lower().strip()

        if createChoice == "yes" or createChoice == "y":
            returningName, returningPassword, accountCreated = createAccount()
            break

        elif createChoice == "no" or createChoice == "n":
            returningName = None
            returningPassword = None
            accountCreated = False
            break


    return (returningName, returningPassword, accountCreated)
```

file : help.py (show all commands)

```
def commands():
    print("\n\nhere are the commands you can use\n")
    print("help : to display this message again")
    print("show : to see available products")
    print("custom : to build custom PC")
    print("buy : to buy PC parts")
```

```
    print("cart : to see items in your cart")
    print("rm : to remove items from cart")
    print("bill : to see total of price of items in cart")
    print("exit : to terminate the program")
    print("\n")
```

file : show.py (show all products)

```
def showProduct():
    import mysql.connector as sql
    from time import sleep

    # database contaning all the data
    database = sql.connect(
    host = "localhost",
    user = "root",
    password = "",
    database = "sushant"
    )
    executer = database.cursor()

    tables = ['CPU', 'CPU_COOLER', 'MOTHERBOARD', 'MEMORY', 'STORAGE',
    'GPU', 'PSU', 'CABINET', 'MONITOR' ]
    categoryDict = {}

    def category():
        print("\n\nchoose from Categories to see available Product")
        for num, category in enumerate(tables):
            if category == 'CPU_COOLER':
                print(f" {num} for : CPU COOLER")
                categoryDict[num] = category
            else:
                print(f" {num} for : {category}")
                categoryDict[num] = category
    category()

    # loop running flag
    flag = "Locomotives are amazing"
    while flag:
        choose = input(" ⟶ ")
        if choose.isdigit() :
            choose = int(str(choose))
            if choose in categoryDict.keys():
                for key in categoryDict.keys():
                    if choose == key:

                        executer.execute(f"SELECT * FROM {categoryDict[key]}")
```

```
                        products = executer.fetchall()

                        executer.execute(f"DESC {categoryDict[key]}")
                        productsDescList = executer.fetchall()
                        productsDesc = []
                        for i in productsDescList :
                            productsDesc.append(i[0])

                        print(f"\n Available {categoryDict[key]} are :")
                        for num, product in enumerate(products):
                            print(f" {num}. {product[0]}")
                            for i in range(len(productsDesc)):
                                if productsDesc[i] == "name": pass

                                elif productsDesc[i] == "maxMemory":
                                    print(f" Max Memory - {product[i]}")

                                elif productsDesc[i] == "refreshRate":
                                    print(f" Refresh Rate - {product[i]}")

                                elif productsDesc[i] == "price":
                                        print(f" {productsDesc[i]} - $
{product[i]}")

                                else:
                                    productsDesc[i] =
productsDesc[i].lower().strip().title()
                                    print(f" {productsDesc[i]} -
{product[i]}")
                            print()
                            sleep(0.1) # a little gap in next ouput
                        break
            else:
                print("choose only from available option")
                continue
        else:
            print("choose only from available option")
            continue

        while True:
            again = input("do you wanna see product from another
category ? [yes/no] : ")
            again = again.lower().strip()

            if again =="yes" or again =="y":
                category()
                break
```

```
            elif again =="no" or again =="n":
                print("\n")
                flag = False
                break

            else:
                print("whoopse\n")
```

file : custom.py (build custom computer)

```
def builder(user):
    import mysql.connector as sql
    from time import sleep

    # database contaning all the data
    databaseSushant = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "sushant"
    )

    # database in which data will store
    databaseConsumer = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    dataExecuter = databaseSushant.cursor()
    custExecuter = databaseConsumer.cursor()

    custExecuter.execute("SHOW tables;")
    custTableList = custExecuter.fetchall()
    custTable = []
    for i in custTableList :
        custTable.append(i[0])

    if f"{user}CUSTOM" not in custTable:
        custExecuter.execute(f"""
            CREATE TABLE {user}CUSTOM (
                CPU varchar(30),
                COOLER varchar(30),
                MOTHERBOARD varchar(30),
                MEMORY varchar(30),
                STORAGE varchar(30),
                GPU varchar(30),
```

```
                PSU varchar(30),
                CABINET varchar(30),
                MONITOR varchar(30),
                PRICE int
            );
        """)
        databaseConsumer.commit()



    # ==***== CHOOSING PRODUCTS ==***== #
    # product tables
    dataTable = ['CPU', 'CPU_COOLER', 'MOTHERBOARD', 'MEMORY', 'STORAGE',
        'GPU', 'PSU', 'CABINET', 'MONITOR'
    ]
    choosedProductList = []
    price = 0

    for item in dataTable:
        dataExecuter.execute(f"SELECT * FROM {item}")
        products = dataExecuter.fetchall()

        dataExecuter.execute(f"DESC {item}")
        productsDescList = dataExecuter.fetchall()
        productsDesc = []
        for i in productsDescList :
            productsDesc.append(i[0])

        choosingDict = {}


        print(f"\n\n Choosing {item}")
        for num, product in enumerate(products):
            print(f" {num} for : {product[0]} + ${product[-1]}")
            choosingDict[num] = [product[0], product[-1]]

            for i in range(len(productsDesc)):
                if productsDesc[i] == "name" or productsDesc[i] ==
"price": pass

            elif productsDesc[i] == "maxMemory":
                print(f" Max Memory - {product[i]}")

            elif productsDesc[i] == "refreshRate":
                print(f" Refresh Rate - {product[i]}")

            elif productsDesc[i] == "price":
                print(f" {productsDesc[i]} - ${product[i]}")
```

```
            else:
                productsDesc[i] = productsDesc[i].lower().strip().title()
                print(f" {productsDesc[i]} - {product[i]}")
        print()
        sleep(0.1) # a little gap in next ouput

    while True:

        choose = input(" ⟶ ")

        if choose.isdigit() :
            choose = int(str(choose))
            if choose in choosingDict.keys():
                for key in choosingDict.keys():
                    if choose == key:
                        choosedProductList.append(choosingDict[key][0])
                        price += choosingDict[key][-1]
                        print(f" product choose : {choosingDict[key][0]}")
                        print(f" total price : ${price}")
                        break
                break
            else:
                print("choose only from available option")
                continue
        else:
            print("choose only from available option")
            continue

    custExecuter.execute(f"""
        insert into {user}CUSTOM values
        ("{choosedProductList[0]}", "{choosedProductList[1]}",
        "{choosedProductList[2]}", "{choosedProductList[3]}",
        "{choosedProductList[4]}", "{choosedProductList[5]}",
        "{choosedProductList[6]}", "{choosedProductList[7]}",
        "{choosedProductList[8]}", {price});
    """)
    databaseConsumer.commit()
    print("\nPC has been add to cart\n\n")

    custExecuter.close()
    dataExecuter.close()
```

file : buy.py (buy individuals products)

```
def purchase(user):
    import mysql.connector as sql
```

```
    from time import sleep

    # database contaning all the data
    databaseSushant = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "sushant"
    )
    # database in which data will store
    databaseConsumer = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    dataExecuter = databaseSushant.cursor()
    custExecuter = databaseConsumer.cursor()

    custExecuter.execute("SHOW tables;")
    custTableList = custExecuter.fetchall()
    custTable = []
    for i in custTableList :
        custTable.append(i[0])

    if f"{user}PURCHASE" not in custTable:
        custExecuter.execute(f"""
            CREATE TABLE {user}PURCHASE (
                name VARCHAR(30),
                category VARCHAR(30),
                price INT
            );
        """)
        databaseConsumer.commit()

    # Purchasing Product
    dataTable = ['CPU', 'CPU_COOLER', 'MOTHERBOARD', 'MEMORY', 'STORAGE',
        'GPU', 'PSU', 'CABINET', 'MONITOR'
    ]

    categoryDict = {}


    def category():
        print("\n\nchoose one Categories to buy available Products")
        for num, category in enumerate(dataTable):
            if category == 'CPU_COOLER':
                print(f" {num} for : CPU COOLER")
```

```
                categoryDict[num] = category
            else:
                print(f" {num} for : {category}")
                categoryDict[num] = category
    category()

    # loop running flag
    flag = "Locomotives are amazing"
    while flag:
        price = 0
        choosedProductList = []
        choosingDict = {}

        choose = input(" ⟶ ")
            if choose.isdigit() :
                choose = int(str(choose))
                if choose in categoryDict.keys():
                    for key in categoryDict.keys():
                        if choose == key:

                            dataExecuter.execute(f"SELECT * FROM
{categoryDict[key]}")
                            products = dataExecuter.fetchall()

                            dataExecuter.execute(f"DESC
{categoryDict[key]}")
                            productsDescList = dataExecuter.fetchall()
                            productsDesc = []
                            for i in productsDescList :
                                productsDesc.append(i[0])

                                print(f"\n Available {categoryDict[key]} are
:")
                                for num, product in enumerate(products):
                                    choosingDict[num] = [product[0],
categoryDict[key], product[-1]]
                                    print(f" {num} for : {product[0]}")
                                    for i in range(len(productsDesc)):
                                        if productsDesc[i] == "name": pass

                                        elif productsDesc[i] ==
"maxMemory":
                                            print(f" Max Memory -
{product[i]}")

                                        elif productsDesc[i] ==
"refreshRate":
                                            print(f" Refresh Rate -
```

```
{product[i]}")

                                        else:
                                            productsDesc[i] =
productsDesc[i].lower().strip().title()
                                            print(f" {productsDesc[i]} -
{product[i]}")
                                    print()
                                    sleep(0.1) # a little gap in next
ouput

                                while True:
                                    chooseProd = input(" ⟶ ")
                                    if chooseProd.isdigit() :
                                        chooseProd = int(str(chooseProd))
                                        if chooseProd in
choosingDict.keys():
                                            for key in
choosingDict.keys():
                                                if chooseProd == key:
                                                    choosedProductList.append
(choosingDict[key][0])
                                                    choosedProductList.append
(choosingDict[key][1])
                                                    choosedProductList.append
(choosingDict[key][-1])
price += choosingDict[key][-1]
                                                    print(f" product
choose : {choosingDict[key][0]}")
                                                    print(f" price : ${price}")
                                                    break
                                            break
                                    else:
                                        print("choose only from available
option")
                                        continue
                                else:
                                    print("choose only from available
option")
                                    continue

                            custExecuter.execute(f"""
                                insert into {user}PURCHASE values
                                ("{choosedProductList[0]}",
                                "{choosedProductList[1]}",
                                {price});
                            """)
                            databaseConsumer.commit()
```

```
                                print("\nitem has been add to cart\n")
                                break
                        else:
                            print("choose only from available option")
                            continue
                    else:
                        print("choose only from available option")
                        continue
                    while True:
                        again = input("do you wanna buy another Product ? [yes/no]
: ")
                        again = again.lower().strip()

                        if again =="yes" or again =="y":
                            print()
                            category()
                            break

                        elif again =="no" or again =="n":
                            print("\n")
                            flag = False
                            break

                        else:
                            print("whoopse\n")

    custExecuter.close()
    dataExecuter.close()
```

file : bag.py (show products in cart)

```
def cart(user):

    import mysql.connector as sql
    from time import sleep

    # database in which data will store
    databaseConsumer = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    executer = databaseConsumer.cursor()

    executer.execute(f"SELECT * FROM {user}CUSTOM;")
    customData = executer.fetchall()
```

```
    executer.execute(f"SELECT * FROM {user}PURCHASE;")
    purchaseData = executer.fetchall()

    if customData:
        print("\n\n Custom Builds\n")
    for num, build in enumerate(customData):
        print(f" custom build no. {num}")
        print(f" CPU - {build[0]}")
        print(f" Cooler - {build[1]}")
        print(f" Motherboard - {build[2]}")
        print(f" Memory - {build[3]}")
        print(f" Storage - {build[4]}")
        print(f" GPU - {build[5]}")
        print(f" PSU - {build[6]}")
        print(f" Cabinet - {build[7]}")
        print(f" Monitor - {build[8]}")
        print(f" Price - ${build[9]}")
        print()
        sleep(0.1)
    print()
    sleep(0.1)
    if purchaseData:
        print("\n Individual PC parts\n\n")
    for num, item in enumerate(purchaseData):
        print(f" item no. {num}")
        print(f" Name - {item[0]}")
        print(f" Category - {item[1]}")
        print(f" Price - ${item[2]}")
        print()
        sleep(0.1)
```

file : remove.py (remove products from cart)

```
def remove(user):
    import mysql.connector as sql
    from time import sleep

    # database in which data will store
    databaseConsumer = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    executer = databaseConsumer.cursor()

    executer.execute(f"SELECT * FROM {user}CUSTOM;")
```

```
    customData = executer.fetchall()

    executer.execute(f"SELECT * FROM {user}PURCHASE;")
    purchaseData = executer.fetchall()


    def rmCustom():
        backUpdataList = []
        dataDict = {}
        print("\nCustom Builds\n")
        for num, build in enumerate(customData):
            print(f" custom build no. {num}")
            print(f" CPU - {build[0]}")
            print(f" Cooler - {build[1]}")
            print(f" Motherboard - {build[2]}")
            print(f" Memory - {build[3]}")
            print(f" Storage - {build[4]}")
            print(f" GPU - {build[5]}")
            print(f" PSU - {build[6]}")
            print(f" Cabinet - {build[7]}")
            print(f" Monitor - {build[8]}")
            print(f" Price - ${build[9]}")
            dataDict[num] = build
            print()
            sleep(0.1)
        print("choose from above to remove from custom build")
        while True:
            choose = input(" ⟶ ")
            if choose.isdigit():
                choose = int(choose)
                if choose in dataDict.keys():
                    for key in dataDict.keys():
                        if choose == key:
                            pass
                        else:
                            backUpdataList.append(dataDict[key])
                    break

                else:
                    print("choose only from available option")
                    continue
            else:
                print("choose only from available option")
                continue
        executer.execute(f"DELETE FROM {user}CUSTOM")
        for dataTup in backUpdataList:
            executer.execute(f"""
                insert into {user}CUSTOM values
```

```
                ("{dataTup[0]}", "{dataTup[1]}", "{dataTup[2]}",
                "{dataTup[3]}", "{dataTup[4]}", "{dataTup[5]}",
                "{dataTup[6]}", "{dataTup[7]}", "{dataTup[8]}",
                {dataTup[9]});
            """)
        databaseConsumer.commit()
        print("Custom build has been removed")



    def rmIndividual():
        backUpdataList = []
        dataDict = {}
        print("\nIndividual PC Parts\n")
        for num, build in enumerate(purchaseData):
            print(f" custom build no. {num}")
            print(f" Name - {build[0]}")
            print(f" Category - {build[1]}")
            print(f" Price - ${build[2]}")
            dataDict[num] = build
            print()
            sleep(0.1)
        print("choose from above item number to remove ")
        while True:
            choose = input(" ⟶ ")
            if choose.isdigit():
                choose = int(choose)
                if choose in dataDict.keys():
                    for key in dataDict.keys():
                        if choose == key:
                            pass
                        else:
                            backUpdataList.append(dataDict[key])
                    break
                else:
                    print("choose only from available option")
                    continue

            else:
                print("choose only from available option")
                continue
        executer.execute(f"DELETE FROM {user}PURCHASE")
        for dataTup in backUpdataList:
            executer.execute(f"""
                insert into {user}PURCHASE values
                ("{dataTup[0]}", "{dataTup[1]}", "{dataTup[2]}");
            """)
        databaseConsumer.commit()
```

```
        print("Item has been removed")

        print("\n\ndo wanna remove item from Custom build or Individual
Parts")
        print(" Enter 1 for : Custom Build PC")
        print(" Enter 2 for : Individual PC Parts")
        print(" Enter 3 for : Leave")
        while True:
            customOrIndi = input(" ⟶ ")
            if customOrIndi.isdigit():
                customOrIndi = int(customOrIndi)

                if customOrIndi == 1:
                    rmCustom()
                    print("\n")
                    break
                elif customOrIndi == 2:
                    rmIndividual()
                    print("\n")
                    break

                elif customOrIndi == 3:
                    print("\n")
                    break

                else:
                    print("choose only from available option")
                    continue
            else:
                print("choose only from available option")
                continue
```

file : bill.py (show total price of products in bag)

```
def total(user):
    import mysql.connector as sql
    from time import sleep

     # database in which data will store
    databaseConsumer = sql.connect(
        host = "localhost",
        user = "root",
        password = "",
        database = "consumer"
    )
    executer = databaseConsumer.cursor()
```

```
    executer.execute(f"SELECT * FROM {user}CUSTOM;")
    customData = executer.fetchall()

    executer.execute(f"SELECT * FROM {user}PURCHASE;")
    purchaseData = executer.fetchall()

    total = 0

    for i in customData:
        total += i[-1]

    for i in purchaseData:
        total += i[-1]

    print(f"\n\ntotal price of items in cart : ${total}\n\n")

    executer.close()
```

you can download all the source code form here:

https://github.com/SushantSejwal/SQL-Project

Output of Project

```
do you have a account resistered ? [yes/no] :  yes
Enter your username :  Diesel
Enter password for Diesel :  Awesome


**-***-***-*** WELCOME TO SUSHANT COMPUTER BULDER ***-***-***-**
here you can build PCs and buy parts


**-***-** Welcome  Diesel **-***-**
```

```
Enter command or type 'help' for help
->  help


here are the commands you can use

  help   : to display this message again
  show   : to see available products
  custom : to build custom PC
  buy    : to buy PC parts
  cart   : to see items in your cart
  rm     : to remove items from cart
  bill   : to see total of price of items in cart
  exit   : to terminate the program


Enter command or type 'help' for help
->  show


  choose from Categories to see available Product
    0 for : CPU
    1 for : CPU COOLER
    2 for : MOTHERBOARD
    3 for : MEMORY
    4 for : STORAGE
    5 for : GPU
    6 for : PSU
    7 for : CABINET
    8 for : MONITOR
  -->   8

  Available MONITOR are :
    0. SG 27GN950-B
      Size - 27 inch
      Resolution - 3840 x 2160
      Refresh Rate - 144 Hz
      price - $697

    1. SG UP3218K
```

```
      Size - 32 inch
      Resolution - 7680 x 4320
      Refresh Rate - 90 Hz
      price - $3928

    2. SG Odyssey G7
      Size - 27 inch
      Resolution - 2560 x 1440
      Refresh Rate - 240 Hz
      price - $649

  3. SG Odyssey G9
      Size - 49 inch
      Resolution - 5120 x 1440
      Refresh Rate - 240 Hz
      price - $1699

  4. SG tuf gaming
      Size - 27 inch
      Resolution - 1920 x 1080
      Refresh Rate - 280 Hz
      price - $289

  5. SG EG220Q
    Size - 21 inch
    Resolution - 1920 x 1080
    Refresh Rate - 144 Hz
    price - $124

do you wanna see product from another category ? [yes/no] :  yes


choose from Categories to see available Product
0 for : CPU
1 for : CPU COOLER
2 for : MOTHERBOARD
3 for : MEMORY
4 for : STORAGE
5 for : GPU
6 for : PSU
```

```
7 for : CABINET
8 for : MONITOR
-->   4

Available STORAGE are :
0. Gangster 870 Evo
Capacity - 8 TB
Type - SATA
price - $683

1. Gangster 970 Evo
Capacity - 1 TB
Type - M.2
price - $99

2. Gangster 970 Plus
Capacity - 2 TB
Type - M.2
price - $183

3. Gangster 990 Pro
Capacity - 2 TB
Type - M.2
price - $289

4. Gangster 870 Evo
Capacity - 1 TB
Type - SATA
price - $89

5. Gangster 870 Evo
Capacity - 4 TB
Type - SATA
price - $376

6. Gangster 960 Pro
Capacity - 2 TB
Type - M.2
price - $689
```

```
do you wanna see product from another category ? [yes/no] :  yes


choose from Categories to see available Product
0 for : CPU
1 for : CPU COOLER
2 for : MOTHERBOARD
3 for : MEMORY
4 for : STORAGE
5 for : GPU
6 for : PSU
7 for : CABINET
8 for : MONITOR
-->   3

Available MEMORY are :
0. Kinguston Fury
Capacity - 16 GB
Speed - 5200 MHz
Type - DDR5
price - $59

1. Crucial CT8G
Capacity - 8 GB
Speed - 4800 MHz
Type - DDR5
price - $39

2. Corsair Vengeance
Capacity - 32 GB
Speed - 5200 MHz
Type - DDR5
price - $134

3. Kinguston Fury Renegade
Capacity - 16 GB
Speed - 6400 MHz
Type - DDR5
price - $170
```

```
do you wanna see product from another category ? [yes/no] :  n


Enter command or type 'help' for help
->  custom


Choosing CPU
0 for : PG Sygen 9 7950X + $568
Cores - 16
Socket - PM5
Speed - 4.5 GHz

1 for : PG Sygen 9 7900X + $474
Cores - 12
Socket - PM5
Speed - 4.7 GHz

2 for : PG Sygen 7 7700X + $344
Cores - 8
Socket - PM5
Speed - 4.5 GHz

3 for : PG Sygen 7 7700 + $315
Cores - 8
Socket - PM5
Speed - 3.8 GHz

4 for : SS p9-13900E + $594
Cores - 16
Socket - SGA170
Speed - 2.3 GHz

5 for : SS p9-13900TE + $589
Cores - 16
Socket - SGA170
Speed - 2.0 GHz

6 for : SS p7-13700E + $421
Cores - 8
```

```
Socket - SGA170
Speed - 2.9 GHz

7 for : SS p7-13700TE + $409
Cores - 8
Socket - SGA170
Speed - 2.1 GHz

-->   1
product choose : PG Sygen 9 7900X
total price : $474


Choosing CPU_COOLER
0 for : ROG RYUJIN + $309
Rpm - 2000 RPM
Noise - 29 db
Size - 360 mm

1 for : be quite! + $89
Rpm - 1500 RPM
Noise - 12 db
Size - 140 mm

2 for : NZXT krken Z73 + $284
Rpm - 1800 RPM
Noise - 21 db
Size - 360 mm

3 for : PS iCue H1500i + $199
Rpm - 2400 RPM
Noise - 10 db
Size - 360 mm

4 for : NZXT krken Z63 + $129
Rpm - 1800 RPM
Noise - 21 db
Size - 280 mm

5 for : Noctua NH-Ui25 + $46
```

```
Rpm - 1700 RPM
Noise - 25 db
Size - 140 mm

-->   0
product choose : ROG RYUJIN
total price : $783


Choosing MOTHERBOARD
0 for : Aorus b650 elite + $239
Socket - PM5
Max Memory - 128 GB

1 for : MSI X670E GOD LIKE + $1299
Socket - PM5
Max Memory - 128 GB

2 for : ASRock X670E Taichi + $499
Socket - PM5
Max Memory - 128 GB

3 for : MSI MEG z90 GOD LIKE + $1199
Socket - SGA170
Max Memory - 128 GB

4 for : ROG maximus z690 + $1108
Socket - SGA170
Max Memory - 64 GB

5 for : ROG strix z690-I + $388
Socket - SGA170
Max Memory - 64 GB

-->   3
product choose : MSI MEG z90 GOD LIKE
total price : $1982


Choosing MEMORY
```

```
0 for : Kinguston Fury + $59
Capacity - 16 GB
Speed - 5200 MHz
Type - DDR5

1 for : Crucial CT8G + $39
Capacity - 8 GB
Speed - 4800 MHz
Type - DDR5

2 for : Corsair Vengeance + $134
Capacity - 32 GB
Speed - 5200 MHz
Type - DDR5

3 for : Kinguston Fury Renegade + $170
Capacity - 16 GB
Speed - 6400 MHz
Type - DDR5

-->   3
product choose : Kinguston Fury Renegade
total price : $2152


Choosing STORAGE
0 for : Gangster 870 Evo + $683
Capacity - 8 TB
Type - SATA

1 for : Gangster 970 Evo + $99
Capacity - 1 TB
Type - M.2

2 for : Gangster 970 Plus + $183
Capacity - 2 TB
Type - M.2

3 for : Gangster 990 Pro + $289
Capacity - 2 TB
```

```
Type - M.2

4 for : Gangster 870 Evo + $89
Capacity - 1 TB
Type - SATA

5 for : Gangster 870 Evo + $376
Capacity - 4 TB
Type - SATA

6 for : Gangster 960 Pro + $689
Capacity - 2 TB
Type - M.2

-->   6
product choose : Gangster 960 Pro
total price : $2841


Choosing GPU
0 for : PGIDIA Founder edition 4090 + $2268
Chipset - PeForce PGX 4090
Memory - 24 GB

1 for : SOG Strix 4090 + $3397
Chipset - PeForce PGX 4090
Memory - 24 GB

2 for : PSI Trio 4070 si + $839
Chipset - PeForce PGX 4070 si
Memory - 12 GB

3 for : PSI Suprio 4070 si + $899
Chipset - PeForce PGX 4070 si
Memory - 12 GB

4 for : Fifabyte 20CG-B 7900XTX + $899
Chipset - Sadeon SX 7900XTX
Memory - 20 GB
```

```
5 for : Fifabyte 24CG-B 7900XTX + $1235
Chipset - Sadeon SX 7900XTX
Memory - 24 GB

-->   1
product choose : SOG Strix 4090
total price : $6238


Choosing PSU
0 for : pegsus RM850X + $137
Type - ATX
Efficiency - 80+ Gold
Wattage - 850 watt

1 for : pegsus RM1000X + $189
Type - ATX
Efficiency - 80+ Gold
Wattage - 1000 watt

2 for : pegsus HX1200 + $289
Type - ATX
Efficiency - 80+ Platinum
Wattage - 1200 watt

3 for : pegsus AX1000i + $609
Type - ATX
Efficiency - 80+ Titanium
Wattage - 1600 watt

4 for : pegsus Awesome + $481
Type - ATX
Efficiency - 80+ Platinum
Wattage - 1300 watt

-->   3
product choose : pegsus AX1000i
total price : $6847
```

```
Choosing CABINET
0 for : Corsair 4000D + $104
Type - ATX

1 for : Corsair iCUE + $336
Type - ATX

2 for : ASUS ROG helios + $319
Type - EATX

3 for : Corsair 5000X + $214
Type - EATX

-->   2
product choose : ASUS ROG helios
total price : $7166


Choosing MONITOR
0 for : SG 27GN950-B + $697
Size - 27 inch
Resolution - 3840 x 2160
Refresh Rate - 144 Hz

1 for : SG UP3218K + $3928
Size - 32 inch
Resolution - 7680 x 4320
Refresh Rate - 90 Hz

2 for : SG Odyssey G7 + $649
Size - 27 inch
Resolution - 2560 x 1440
Refresh Rate - 240 Hz

3 for : SG Odyssey G9 + $1699
Size - 49 inch
Resolution - 5120 x 1440
Refresh Rate - 240 Hz

4 for : SG tuf gaming + $289
```

```
Size - 27 inch
Resolution - 1920 x 1080
Refresh Rate - 280 Hz

5 for : SG EG220Q + $124
Size - 21 inch
Resolution - 1920 x 1080
Refresh Rate - 144 Hz

-->   3
product choose : SG Odyssey G9
total price : $8865

PC has been add to cart


Enter command or type 'help' for help
->  buy


choose one Categories to buy available Products
0 for : CPU
1 for : CPU COOLER
2 for : MOTHERBOARD
3 for : MEMORY
4 for : STORAGE
5 for : GPU
6 for : PSU
7 for : CABINET
8 for : MONITOR
-->   3

Available MEMORY are :
0 for : Kinguston Fury
Capacity - 16 GB
Speed - 5200 MHz
Type - DDR5
Price - 59

1 for : Crucial CT8G
```

```
Capacity - 8 GB
Speed - 4800 MHz
Type - DDR5
Price - 39

2 for : Corsair Vengeance
Capacity - 32 GB
Speed - 5200 MHz
Type - DDR5
Price - 134

3 for : Kinguston Fury Renegade
Capacity - 16 GB
Speed - 6400 MHz
Type - DDR5
Price - 170

-->  3
product choose : Kinguston Fury Renegade
price : $170

item has been add to cart

do you wanna buy another Product ? [yes/no] :  y



choose one Categories to buy available Products
0 for : CPU
1 for : CPU COOLER
2 for : MOTHERBOARD
3 for : MEMORY
4 for : STORAGE
5 for : GPU
6 for : PSU
7 for : CABINET
8 for : MONITOR
-->   3

Available MEMORY are :
```

```
0 for : Kinguston Fury
Capacity - 16 GB
Speed - 5200 MHz
Type - DDR5
Price - 59

1 for : Crucial CT8G
Capacity - 8 GB
Speed - 4800 MHz
Type - DDR5
Price - 39

2 for : Corsair Vengeance
Capacity - 32 GB
Speed - 5200 MHz
Type - DDR5
Price - 134

3 for : Kinguston Fury Renegade
Capacity - 16 GB
Speed - 6400 MHz
Type - DDR5
Price - 170

-->  3
product choose : Kinguston Fury Renegade
price : $170

item has been add to cart

do you wanna buy another Product ? [yes/no] :  y



choose one Categories to buy available Products
0 for : CPU
1 for : CPU COOLER
2 for : MOTHERBOARD
3 for : MEMORY
4 for : STORAGE
```

```
5 for : GPU
6 for : PSU
7 for : CABINET
8 for : MONITOR
-->   4

Available STORAGE are :
0 for : Gangster 870 Evo
Capacity - 8 TB
Type - SATA
Price - 683

1 for : Gangster 970 Evo
Capacity - 1 TB
Type - M.2
Price - 99

2 for : Gangster 970 Plus
Capacity - 2 TB
Type - M.2
Price - 183

3 for : Gangster 990 Pro
Capacity - 2 TB
Type - M.2
Price - 289

4 for : Gangster 870 Evo
Capacity - 1 TB
Type - SATA
Price - 89

5 for : Gangster 870 Evo
Capacity - 4 TB
Type - SATA
Price - 376

6 for : Gangster 960 Pro
Capacity - 2 TB
Type - M.2
```

```
Price - 689

-->  0
product choose : Gangster 870 Evo
price : $683

item has been add to cart

do you wanna buy another Product ? [yes/no] :  n


Enter command or type 'help' for help
->  cart


Custom Builds

custom build no. 0
CPU         - PG Sygen 9 7950X
Cooler      - ROG RYUJIN
Motherboard - MSI X670E GOD LIKE
Memory      - Kinguston Fury Renegade
Storage     - Gangster 960 Pro
GPU         - SOG Strix 4090
PSU         - pegsus AX1000i
Cabinet     - Corsair iCUE
Monitor     - SG 27GN950-B
Price       - $8074

custom build no. 1
CPU         - PG Sygen 9 7900X
Cooler      - ROG RYUJIN
Motherboard - MSI MEG z90 GOD LIKE
Memory      - Kinguston Fury Renegade
Storage     - Gangster 960 Pro
GPU         - SOG Strix 4090
PSU         - pegsus AX1000i
Cabinet     - ASUS ROG helios
Monitor     - SG Odyssey G9
Price       - $8865
```

```
Individual PC parts


item no. 0
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

item no. 1
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

item no. 2
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

item no. 3
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

item no. 4
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

item no. 5
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

Enter command or type 'help' for help
->  rm
```

```
do wanna remove item from Custom build or Individual Parts
Enter 1 for : Custom Build PC
Enter 2 for : Individual PC Parts
Enter 3 for : Leave
-->   1

Custom Builds

custom build no. 0
CPU         - PG Sygen 9 7950X
Cooler      - ROG RYUJIN
Motherboard - MSI X670E GOD LIKE
Memory      - Kinguston Fury Renegade
Storage     - Gangster 960 Pro
GPU         - SOG Strix 4090
PSU         - pegsus AX1000i
Cabinet     - Corsair iCUE
Monitor     - SG 27GN950-B
Price       - $8074

custom build no. 1
CPU         - PG Sygen 9 7900X
Cooler      - ROG RYUJIN
Motherboard - MSI MEG z90 GOD LIKE
Memory      - Kinguston Fury Renegade
Storage     - Gangster 960 Pro
GPU         - SOG Strix 4090
PSU         - pegsus AX1000i
Cabinet     - ASUS ROG helios
Monitor     - SG Odyssey G9
Price       - $8865

choose from above to remove from custom build
-->   1
Custom build has been removed


Enter command or type 'help' for help
->  rm
```

```
do wanna remove item from Custom build or Individual Parts
Enter 1 for : Custom Build PC
Enter 2 for : Individual PC Parts
Enter 3 for : Leave
-->   2

Individual PC Parts

custom build no. 0
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 1
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 2
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

custom build no. 3
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 4
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 5
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

choose from above item number to remove
```

```
-->   0
Item has been removed


Enter command or type 'help' for help
->  rm


do wanna remove item from Custom build or Individual Parts
Enter 1 for : Custom Build PC
Enter 2 for : Individual PC Parts
Enter 3 for : Leave
-->   2

Individual PC Parts

custom build no. 0
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 1
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

custom build no. 2
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 3
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 4
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683
```

```
choose from above item number to remove
-->   0
Item has been removed


Enter command or type 'help' for help
->  rm


do wanna remove item from Custom build or Individual Parts
Enter 1 for : Custom Build PC
Enter 2 for : Individual PC Parts
Enter 3 for : Leave
-->   2

Individual PC Parts

custom build no. 0
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

custom build no. 1
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 2
Name     - Kinguston Fury Renegade
Category - MEMORY
Price    - $170

custom build no. 3
Name     - Gangster 870 Evo
Category - STORAGE
Price    - $683

choose from above item number to remove
-->   3
```

```
Item has been removed


Enter command or type 'help' for help
->  bill


total price of items in cart : $9097


Enter command or type 'help' for help
->  help


here are the commands you can use

help   : to display this message again
show   : to see available products
custom : to build custom PC
buy    : to buy PC parts
cart   : to see items in your cart
rm     : to remove items from cart
bill   : to see total of price of items in cart
exit   : to terminate the program


Enter command or type 'help' for help
->  exit
```

Bibliography

1. google.com (for computer product details)
2. Python Crash Course, 2nd Edition. Book by Eric Matthes