VELAMMAL VIDHYASHRAM

MAMBAKKAM



COMPUTER SCIENCE

(PYTHON)

Topic:

Academic year: 2022 - 2023

### Done By:

Kesavram.V.S

XII – A

BONAFIDE CERTIFICATE

This is to certify that this COMPUTER SCIENCE Project on the topic

**………………………………….** has been successfully completed by

**Kesavram.V.S** of class XII ‘A ‘Roll.no…………… at

Velammal Vidhyashram, Mambakkam, for the partial fulfilment of this project as a part of All India Senior School Certificate Examination- CBSE, New Delhi for the academic Year **2022 – 2023 .**

Date: ……………………..

Signature of Principal Signature of the Guide

Signature of the Signature of the

Internal Examiner External Examiner

#### ACKNOWLEDGEMENT:

I wish to express my deep gratitude and sincere thanks to the Senior Principal ……………

………………………. Velammal Vidhyashram, Mambakkam for his encouragement

given to me and for all the facilities that he provided for this project work. I sincerely

appreciate this magnanimity by taking me into his fold for which I shall remain indebted to

him.

I extend my hearty thanks to ………………………….. Computer Science Teacher, who

guided me to the successful completion of this project. I take this opportunity to express my

deep sense of gratitude for her valuable guidance, constant encouragement, immense

motivation, which sustained my efforts at all the stage of this project work.

I would like to thank my lab assistant, lab in charge and technical staffs for providing correct

information which makes the completion of this project work in an excellent way.

I can’t forget to offer my sincere thanks to parents, and also to my classmates who helped

me to carry out this project work successful and for their valuable advice and support, which

received from them time to time.

## TABLE OF CONTENT:

1. ABSTRACT
2. SYSTEM CONFIGURATION
3. INTRODUCTION

PYTHON MYSQL

1. RELATION BETWEEN PYTHON AND MYSQL
2. MYSQL QUERIES USED IN THIS PROJECT
3. DATABASE CONNECTIVITY
4. SOURCE CODE
5. OUTPUT (SCREENSHOT)
6. CONCLUSION
7. BIBILOGRAPHY

**ABSTRACT:**

This software is used to make sql commands easy through python interface.User can just run this program to create modify or delete values from a table in a database selected by them.

**OBJECTIVES OF THE PROJECT:**

The objective of this project is to let the students apply the programming knowledge into a real- world situation/problem and exposed the students how programming skills helps in developing a good software.

1. Write programs utilizing modern software tools.

2. Apply object oriented programming principles effectively when developing small to medium sized projects.

3. Write effective procedural code to solve small to medium sized problems.

4. Students will demonstrate a breadth of knowledge in computer science, as exemplified in the areas of systems, theory and software development.

5. Students will demonstrate ability to conduct a research or applied Computer Science project, requiring writing and presentation skills which exemplify scholarly style in computer science

# SYSTEM CONFIGURATION:

#### HARDWARE CONFIGURATION

##### Microsoft windows 7 professional/windows 8/windows 8.2:

* **Processor :** Intel Core i5 4400U
* **Memory :** 4 GB (32-bit)
* **Disk space :** 512 GB of disk space

#### SOFTWARE REQUIREMENTS

* 1 GB RAM (2 GB+ recommended)
* 9-58 GB free hard disk space depending on edition and configuration, including space required for temporary files
* DVD-ROM drive (if installing from a Media Kit DVD)
* Basic GPU – Any vendor DirectX 9.0 class or better (Pixel Shader Level 2)
* Intel® Pentium® or compatible, 1.6 GHz minimum (2GHz+ recommended)
* 1024x768 or higher-resolution monitor

MOUSE OR OTHER POINTING DEVICE

# INTRODUCTION:

## PYTHON:

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt.sources newsgroup in 1991, and version

* 1. was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of life of January 1,2020 has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do.

Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python’s Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

### Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of programming Language Index.

### Python is Interpreted

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

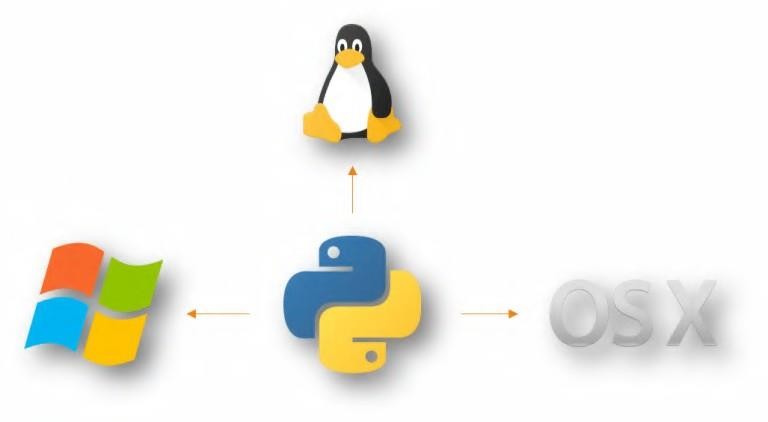
One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

### Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smartphones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.



### Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

### Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

MYSQL:

##### Database Management System & Types of DBMS:

A **Database Management System**(**DBMS**) is a software application that interacts with the user, applications and the database itself to capture and analyze data.The data stored in the database can be modified, retrieved and deleted, and can be of any type like strings, numbers, images etc.

**Types of DBMS:**

There are mainly 4 types of DBMS, which are Hierarchical, Relational, Network, and Object-Oriented DBMS.

Hierarchical DBMS:As the name suggests, this type of DBMS has a style of predecessor- successor type of relationship. So, it has a structure similar to that of a tree, wherein the nodes represent records and the branches of the tree represent fields.

Relational DBMS (RDBMS): This type of DBMS, uses a structure that allows the users to identify and access data *in relation* to another piece of data in the database.

Network DBMS: This type of DBMS supports many to many relations wherein multiple member records can be linked.

Object-oriented DBMS: This type of DBMS uses small individual software called objects. Each object contains a piece of data, and the instructions for the actions to be done with the data.

##### Structured Query Language (SQL)

SQL is the core of a relational database which is used for accessing and managing the database. By using SQL, you can add, update or delete rows of data, retrieve subsets of

information,modify databases and perform many actions. The different subsets of SQL are as follows:

* + - ***DDL*** *(Data Definition Language) –*It allows you to perform various operations on the database such as CREATE, ALTER and DELETE objects.
    - ***DML*** *(Data Manipulation Language)*– It allows you to access and manipulate data. It helps you to insert, update, delete and retrieve data from the database.
    - ***DCL*** *(Data Control Language)*– It allows you to control access to the database. Example – Grant or Revoke access permissions.
    - ***TCL****(Transaction Control Language)* – It allows you to deal with the transaction of the database. Example – Commit, Rollback, Savepoint, Set Transaction.

##### MySQL & its Features

**MySQL** is an open-source relational database management system that works on many platforms. It provides multi-user access to support many storage engines and is backed by Oracle. So,you can buy a commercial license version from Oracle to get premium support services.

The features of MySQL are as follows:**Ease of Management –**The software very easily gets downloaded and also uses an event scheduler to schedule the tasks automatically.

* + - **Robust Transactional Support –**Holds the ACID (Atomicity, Consistency, Isolation, Durability) property, and also allows distributed multi-version support.
    - **Comprehensive Application Development –**MySQL has plugin libraries to embed the database into any application. It also supports stored procedures, triggers, functions, views and many more for application development. You can refer to the ***RDS Tutorial***, to understand Amazon’s RDBMS.
    - **High Performance –**Provides fast load utilities with distinct memory caches and table index partitioning.
    - **Low Total Cost Of Ownership –**This reduces licensing costs and hardware expenditures.
    - **Open Source & 24 \* 7 Support –**This RDBMS can be used on any platform and offers 24\*7 support for open source and enterprise edition.
    - **Secure Data Protection –**MySQL supports powerful mechanisms to ensure that only authorized users have access to the databases.
    - **High Availability –**MySQL can run high-speed master/slave replication configurations and it offers cluster servers.
    - **Scalability & Flexibility –**With MySQL you can run deeply embedded applications and create data warehouses holding a humongous amount of data.

##### MySQL Data Types

* + - **Numeric –** This data type includes integers of various sizes, floating-point(real) of various precisions and formatted numbers.
    - **Character-string –** These data types either have a fixed, or a varying number of characters. This data type also has a variable-length string called *CHARACTER LARGE OBJECT*(*CLOB*) which is used to specify columns that have large text values.
    - **Bit-string –** These data types are either of a fixed length or varying length of bits. There is also a variable-length bit string data type called *BINARY LARGE OBJECT(BLOB),* which is available to specify columns that have large binary values, such as images.
    - **Boolean –**This data type has TRUE or FALSE values. Since SQL, has NULL values, a three-valued logic is used, which is UNKNOWN.
    - **Date & Time –**The DATE data type has: YEAR, MONTH, and DAY in the form YYYY-MM-DD. Similarly, the TIME data type has the components HOUR, MINUTE, and SECOND in the form HH:MM: SS. These formats can change based on the requirement.
    - **Timestamp & Interval –**The TIMESTAMP data type includes a minimum of six positions, for decimal fractions of seconds and an optional WITH TIME ZONE qualifier in addition to the DATE and TIME fields. The INTERVAL data type mentions a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp.

##### Python MySQL Database Connection:

**Arguments required to connect MySQL from Python**

You need to know the following detail of the MySQL server to perform the connection from Python.

* + - **Username** – i.e., the username that you use to work with MySQL Server. The default username for the MySQL database is a **root**
    - **Password** – Password is given by the user at the time of installing the MySQL database. If you are using root then you won’t need the password.
    - **Host Name** – is the server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost, or it’s IP, i.e. 127.0.0.0
    - **Database Name** – Database name to which you want to connect.

##### READ Operation

READ Operation on any database means to fetch some useful information from the database.

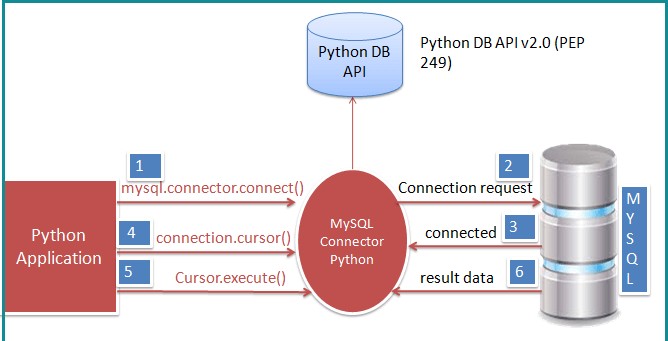
Once our database connection is established, you are ready to make a query into this database. You can use either **fetchone()** method to fetch single record or **fetchall()** method to fetech multiple values from a database table.

* + - * **fetchone()** − It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.
      * **fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.

##### DATABASE CONNECTIVITY:

**Steps to connect MySQL database in Python using MySQL Connector Python**

1. Install MySQL Connector Python using pip.
2. Use the **mysql.connector.connect()**method of MySQL Connector Python with required parameters to connect MySQL.
3. Use the connection object returned by a connect()method to create a **cursor** object to perform Database Operations.
4. The **cursor.execute()**to execute SQL queries from Python.
5. Close the Cursor object using a **cursor.close()** and MySQL database connection using **connection.close()** after your work completes.
6. Catch Exception if any that may occur during this process.



SOURCE CODE:

# Importing the mySQL.connector module to python

import mysql.connector

# Getting the details of connection from the user

host = input("Enter the name of host: ")

user = input("Enter the name of user: ")

passwd = input("Enter the name of the password: ")

# Connecting python to mySQL through the information given by user

mydb = mysql.connector.connect(host = host, user = user , passwd = passwd)

mycur = mydb.cursor()

# Display the database available in the server

database = 'show databases'

mycur.execute(database)

print('The list of databases are: ')

for X in mycur:

    print(X)

# Getting the database

db = input("Enter your database: ")

dbs = 'use {}'.format(db)

mycur.execute(dbs)

# Displaying the tables available in the database

s = 'show tables'

mycur.execute(s)

print('The list of tables in {} are: '.format(db))

for Y in mycur:

    print(Y)

tab = input("Enter the table name: ")

# Implementing a function for a menu driven program

def customize():

    while True:

        print('Menu!!!')

        print("\n 1.Add (To add columns in the selected table) \n 2.Modify (To change the datatype of the selected table) \n 3.Drop columns (To drop any column from the selected table) \n 4.Show (To display values from the selected table) \n 5.Insert (To insert values in the selected table) \n 6.Custom (Acts as a terminal) \n 7.End (To break)")

        x = int(input("Enter your choice: "))

if x == 1:

            y = input("Enter column name to be added in {} table: ".format(tab))

            z = input("Enter the datatype: ")

            p = 'alter table {} add {} {}'.format(tab,y,z)

            mycur.execute(p)

            print("Column '{}' has been sucessfully added to table {}".format(y,tab))

            dis = input("Do you want to display the table(Y for yes): ")

            if dis.lower() == 'y':

                show = 'show columns from {}'.format(tab)

                mycur.execute(show)

                for X in mycur:

                    print(X[0],end =' ')

                disp = 'select \* from {}'.format(tab)

                mycur.execute(disp)

                for Y in mycur:

                    print(Y)

            else:

                continue

            mydb.commit()

        elif x == 2:

            y = input("Enter name of the column to be modified in {} table: ".format(tab))

            z = input("Enter the datatype of column {}: ".format(y))

            p = 'alter table {} modify {} {}'.format(tab,y,z)

            mycur.execute(p)

            print("column '{}' of table '{}' has been changed to '{}' datatype".format(y,tab,z))

            dis = input("Do you want to display the table(Y for yes): ")

            if dis.lower() == 'y':

                show = 'show columns from {}'.format(tab)

                mycur.execute(show)

                for X in mycur:

                    print(X[0],end =' ')

                disp = 'select \* from {}'.format(tab)

                mycur.execute(disp)

                for Y in mycur:

                    print(Y)

            else:

                continue

            mydb.commit()

        elif x == 3:

            y = input("Enter name of the column to be dropped from {} table: ".format(tab))

            p = 'alter table {} drop {}'.format(tab,y)

            mycur.execute(p)

            print("The column '{}' from table '{}' have been removed".format(y,tab))

            dis = input("Do you want to display the table(Y for yes): ")

            if dis.lower() == 'y':

                show = 'show columns from {}'.format(tab)

                mycur.execute(show)

                for X in mycur:

                    print(X[0],end =' ')

                disp = 'select \* from {}'.format(tab)

                mycur.execute(disp)

                for Y in mycur:

                    print(Y)

            else:

                continue

            mydb.commit()

        elif x == 4:

            y = input("Enter the column to display in {} table(\* for all the columns): ".format(tab))

            z = input("Enter the condition: ")

            if z == '':

                p = 'select {} from {}'.format(y,tab,z)

            else:

                p = 'select {} from {} where {}'.format(y,tab,z)

            mycur.execute(p)

            for X in mycur:

                print(X)

            mydb.commit()

        elif x == 5:

            y = input("Enter the names of columns where values are added in {} table(seperated by Commas): ".format(tab))

            z = input("Enter the values of corresponding columns in {} table: ".format(tab))

            p = 'insert into {}({}) values({})'.format(tab,y,z)

            mycur.execute(p)

            print("Value inserted successfully")

            dis = input("Do you want to display the table(Y for yes): ")

            if dis.lower() == 'y':

                show = 'show columns from {}'.format(tab)

                mycur.execute(show)

                for X in mycur:

                    print(X[0],end =' ')

                disp = 'select \* from {}'.format(tab)

                mycur.execute(disp)

                for Y in mycur:

                    print(Y)

            else:

                continue

            mydb.commit()

        elif x == 6:

            y = input('Enter your custom command: ')

            p = '{};'.format(y)

            mycur.execute(p)

            for X in mycur:

                print(X)

            dis = input("Do you want to display the table(Y for yes): ")

            if dis.lower() == 'y':

                show = 'show columns from {}'.format(tab)

                mycur.execute(show)

                for X in mycur:

                    print(X[0],end =' ')

                disp = 'select \* from {}'.format(tab)

                mycur.execute(disp)

                for Y in mycur:

                    print(Y)

else:

                continue

            mydb.commit()

elif x == 7:

            y = input("Enter any column name to be displayed: ")

            p = input("Enter any condition: ")

            q = input("Enter a column to be arranged: ")

            z = input("Enter asc or desc: ")

            if p == "":

                if z == "asc":

                    mycur.execute("select {} from {} order by {}".format(y,tab,q))

                    for i in mycur:

                        print(i)

                elif z == "desc":

                    mycur.execute("select {} from {} order by {} desc".format(y,tab,q))

                    for j in mycur:

                        print(j)

            elif p != "":

                if z == "asc":

                    mycur.execute("select {} from {} where {} order by {}".format(y,tab,p,q))

                    for k in mycur:

                        print(k)

  elif z == "desc":

                  mycur.execute("select {} from {} where {} order by {} desc".format(y,tab,p,q))

                    for k in mycur:

                        print(k)

mydb.commit()

        elif x == 8:

print(“Thank you for using this program”)

            break

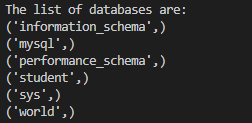
customize()

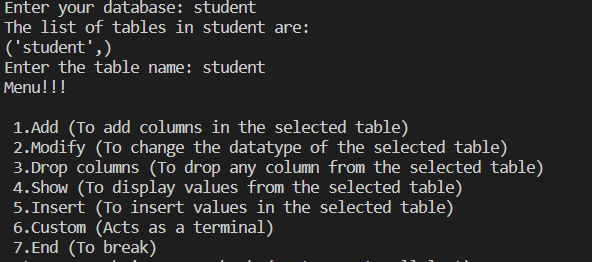
OUTPUT:

Connection of sql and python:

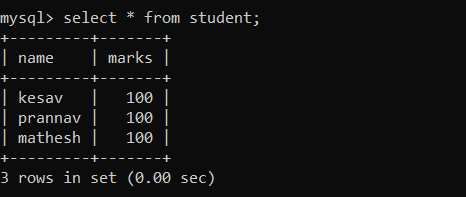


CHOOSING THE DATABASE AND LISTING THE TABLES PRESENT IN IT:

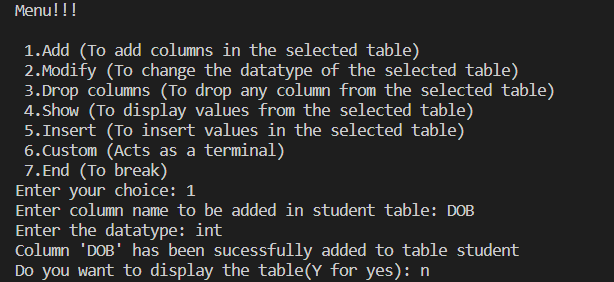




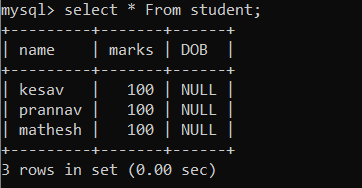
(I)Student table before the operation:



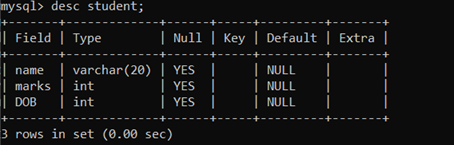
Operation 1: Adding columns to the table student



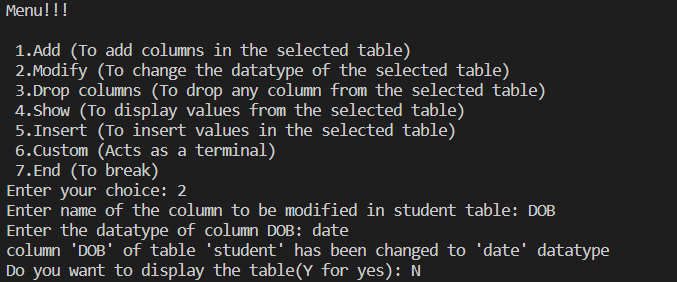
Student table after the operation:



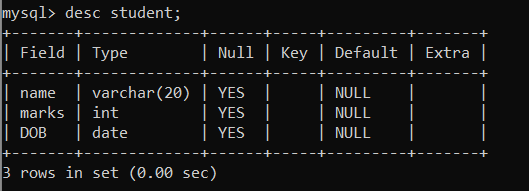
(II)Student table before the operation:



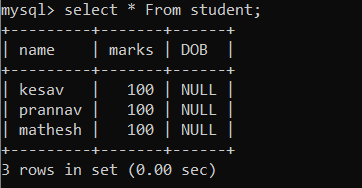
Operation 2: Modifying the column:

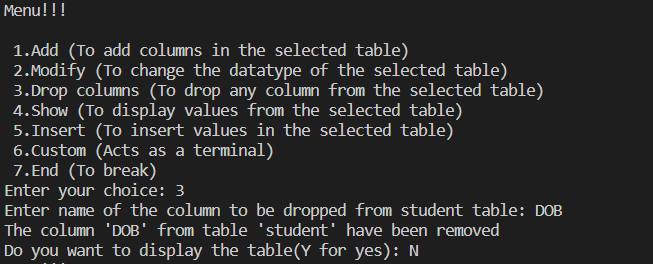


Student table after the operation:

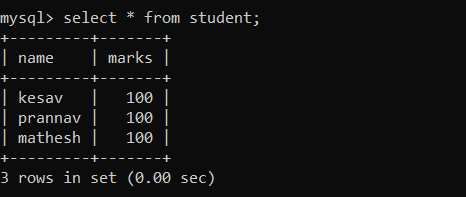


(III) Student table before the operation:

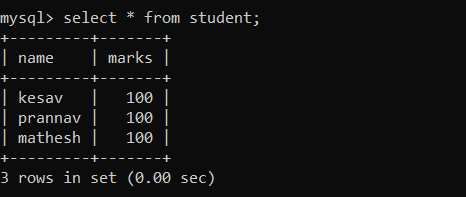


Operation 3: Dropping the column:  


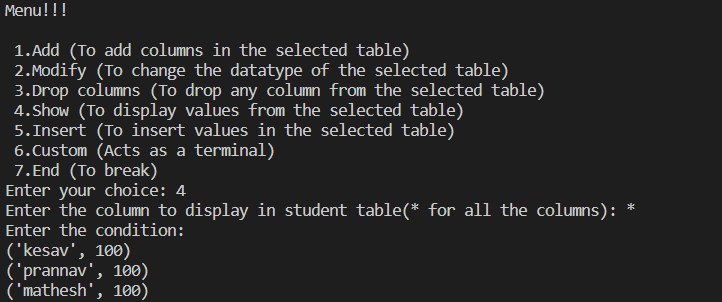
Student table after the operation:



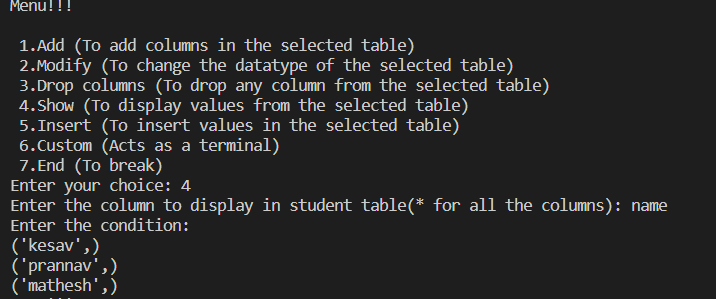
(IV) Student table before the operation:



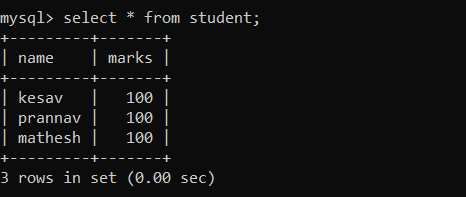
Operation 4 : Displaying columns from the table:

Full table:  


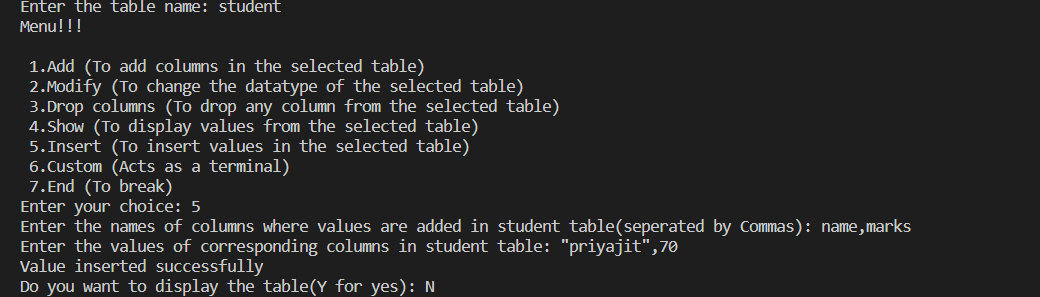
Particular columns of the table:

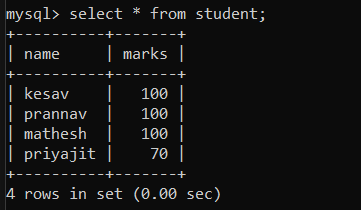


(V) Student table before the operation:

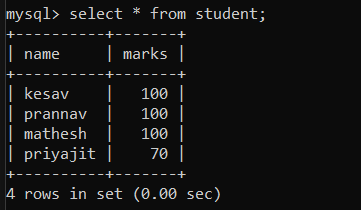


Operation 5: Inserting values into the columns

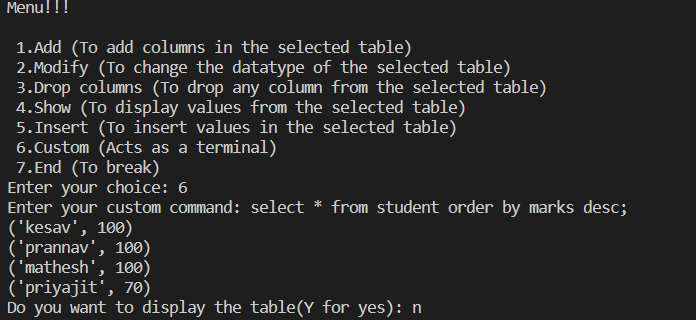


Student table after the operation:  


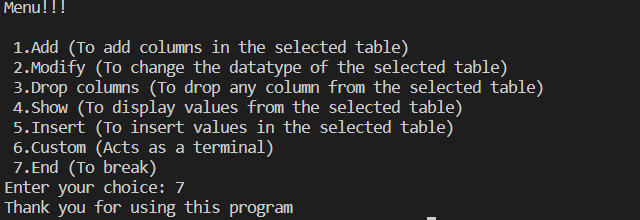
(VI) Student table before the operation:



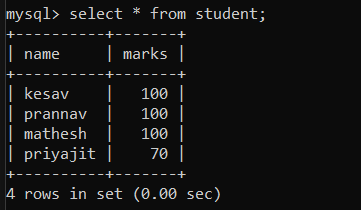
Operation 6: Custom terminal interface in python for custom commands



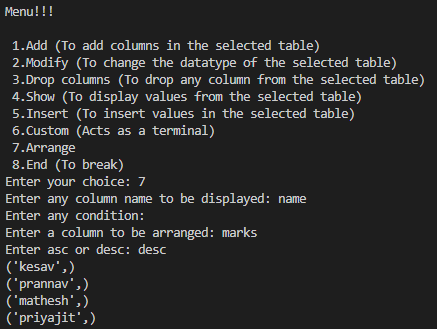
(VII) Breaking the loop:



(VIII) Student table before the operation:



Operation 8:Arranging the table:



BIBLIOGRAPHY:

* CBSE NCERT TEXTBOOK
* SUMITA ARORA CLASS 11
* SUMITA ARORA CLASS 12