Python Database Communication (PDBC)

Every application or project required to store or save data permanently. Application can store data permanently using two systems.

1. File System
2. Database Management System

Limitations of files

1. Files cannot store large of amount data
2. Files are not secured because files are managed by operating system and OS does not provide security for files.
3. File system does not provide any Query language

To overcome above limitations we are using database system.

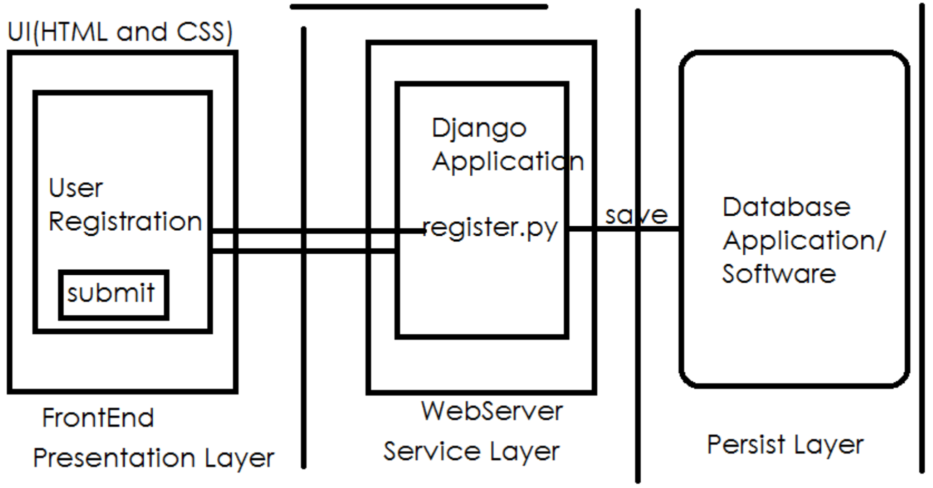
Database System provides the following features

1. Data stored inside database is secured
2. Database can hold large amount data
3. Database system provides Query Language for manipulation of data (SQL).

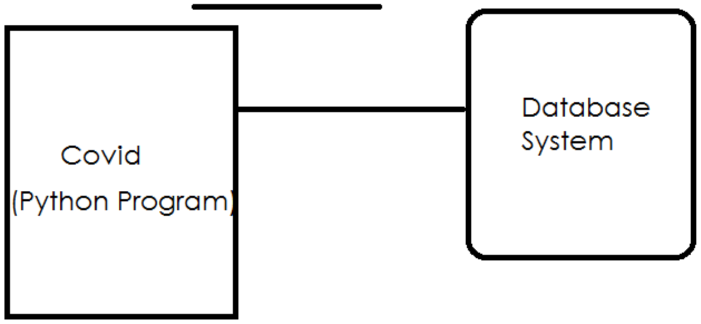
Database applications or Database Software's

1. Oracle
2. Mysql
3. SQLServer
4. MangoDB
5. DB2
6. MSAccess
7. MSExcel
8. Sybase

Web Development



Data Science



SQL

SQL stands for Structured Query Language, is standard which is used to communicate with database or to perform operations on database.
SQL provide set of commands, these command are classified into different categories based on the operations.

1. **DDL → Data Definition Language**
    a. Create
    b. Alter
    c. Drop
2. **DML → Data Manipulation Language**
    a. Insert
    b. Update
    c. Delete
3. **DRL/DQL → Data Retrieval Language**
    a. Select
4. **DCL → Data Control Language**

    a. Grant
    b. Revoke

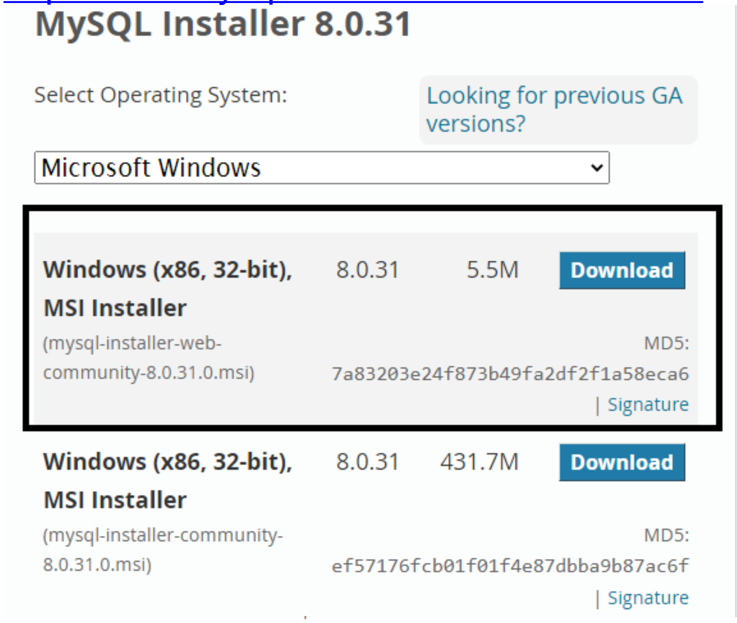
5. **TCL → Transaction Control Language**
    a. Commit
    b. Rollback
    c. Savepoint

MySQL Database

MYSQL Database software is product of Oracle Corp.

How to download and install mysql software

1. https://dev.mysql.com/downloads/installer/



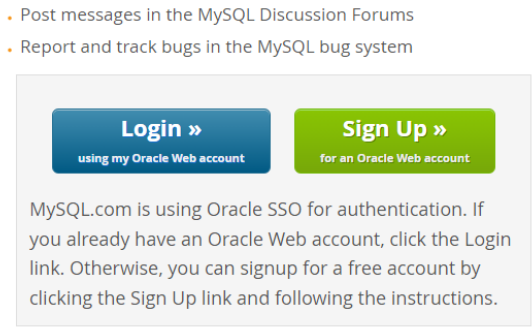
MySQL Installer 8.0.31

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.31.0.msi)	8.0.31	5.5M	Download MD5: 7a83203e24f873b49fa2df2f1a58eca6 | Signature
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.31.0.msi)	8.0.31	431.7M	Download MD5: ef57176fcb01f01f4e87dbba9b87ac6f | Signature

2.

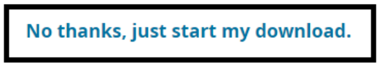


- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login » using my Oracle Web account

Sign Up » for an Oracle Web account

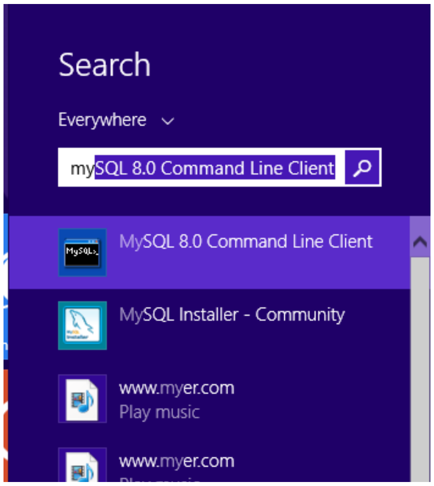
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

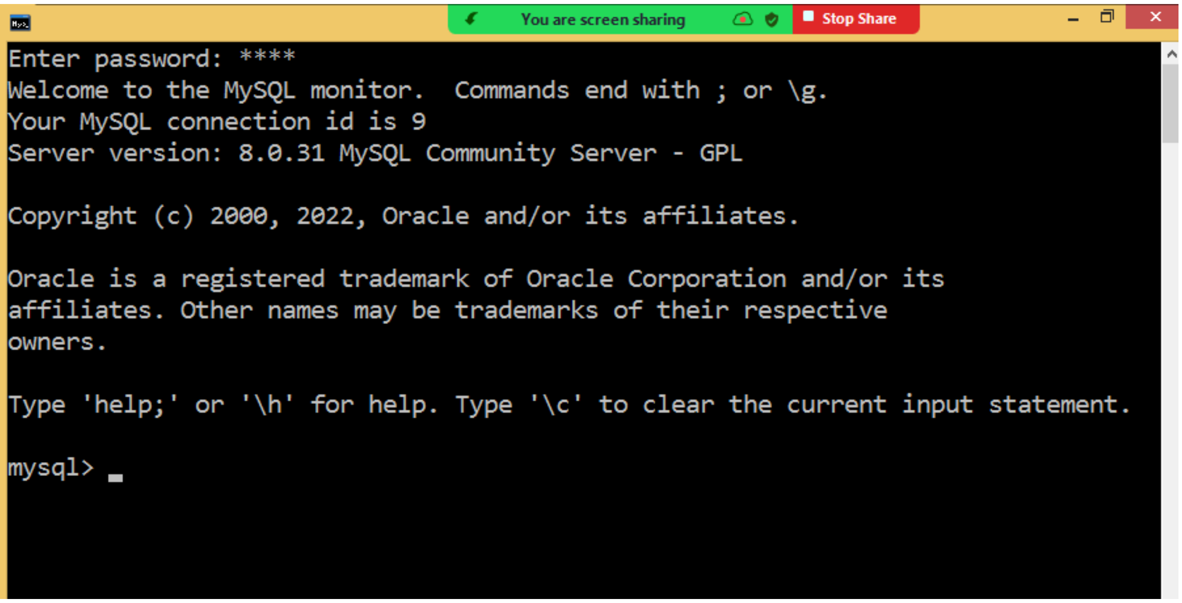
3. 

No thanks, just start my download.

MySQL Command line Client (Tool)

MySQL Commandline client, is command line interface provided by MYSQL to communicate with MYSQL database.


Search
Everywhere
mySQL 8.0 Command Line Client
MySQL 8.0 Command Line Client
MySQL Installer - Community
www.myer.com
Play music
www.myer.com



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

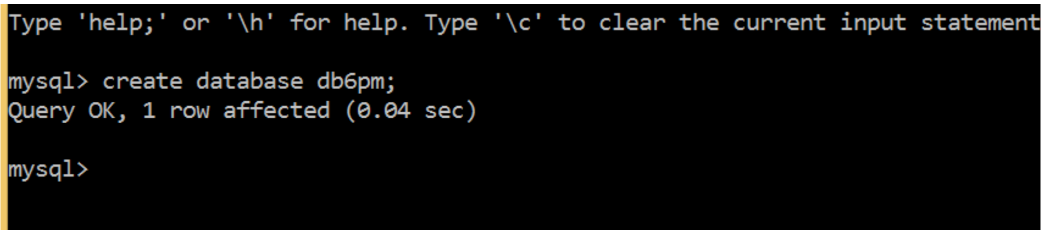
Create Database

Database is a collection of database objects

1. Tables
2. Index
3. Synonyms
4. Cluster

How to create database?

Create database <database-name>;

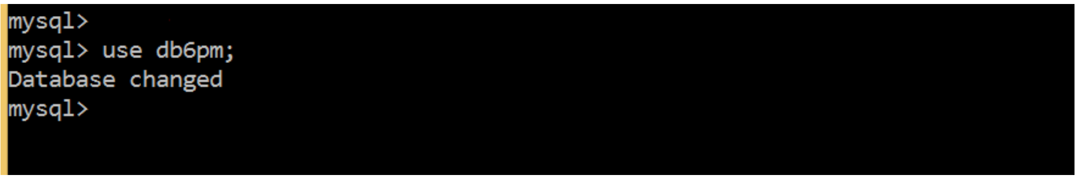


```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

mysql> create database db6pm;
Query OK, 1 row affected (0.04 sec)

mysql>
```

Show databases;

This command display existing databases

use <database-name>
This command is used to open database or activate database



```
mysql>
mysql> use db6pm;
Database changed
mysql>
```

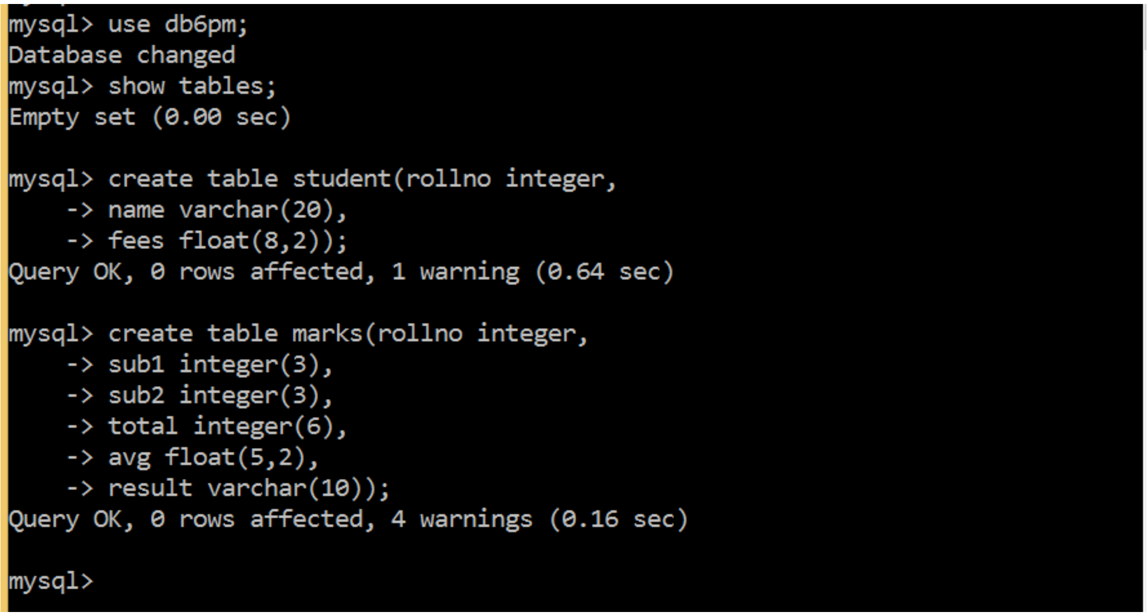
Within database data is stored inside tables. A table is set of rows and columns.

Create table
This command is used for creating table

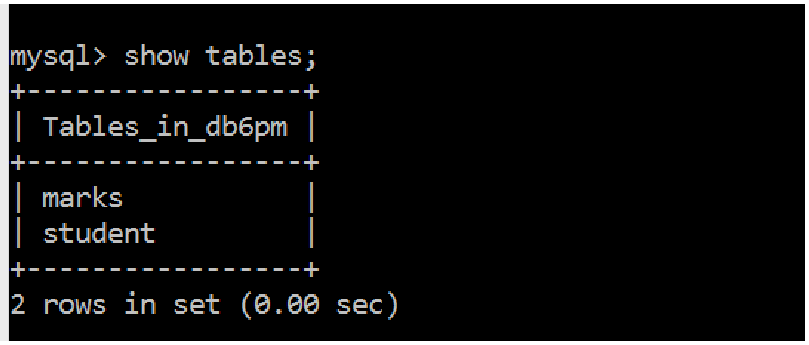
Create table <table-name>(<column-name> <data-type>,
<column-name> <data-type>,
<column-name> <data-type>,…)

MySQL Data types

1. Integer
2. Float
3. Varchar → String
4. Date
5. Time



```
mysql> use db6pm;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table student(rollno integer,
    -> name varchar(20),
    -> fees float(8,2));
Query OK, 0 rows affected, 1 warning (0.64 sec)

mysql> create table marks(rollno integer,
    -> sub1 integer(3),
    -> sub2 integer(3),
    -> total integer(6),
    -> avg float(5,2),
    -> result varchar(10));
Query OK, 0 rows affected, 4 warnings (0.16 sec)

mysql>
```



```
mysql> show tables;
+------------------+
| Tables_in_db6pm |
+------------------+
| marks            |
| student          |
+------------------+
2 rows in set (0.00 sec)
```

DML commands

These commands are used to manipulate data,

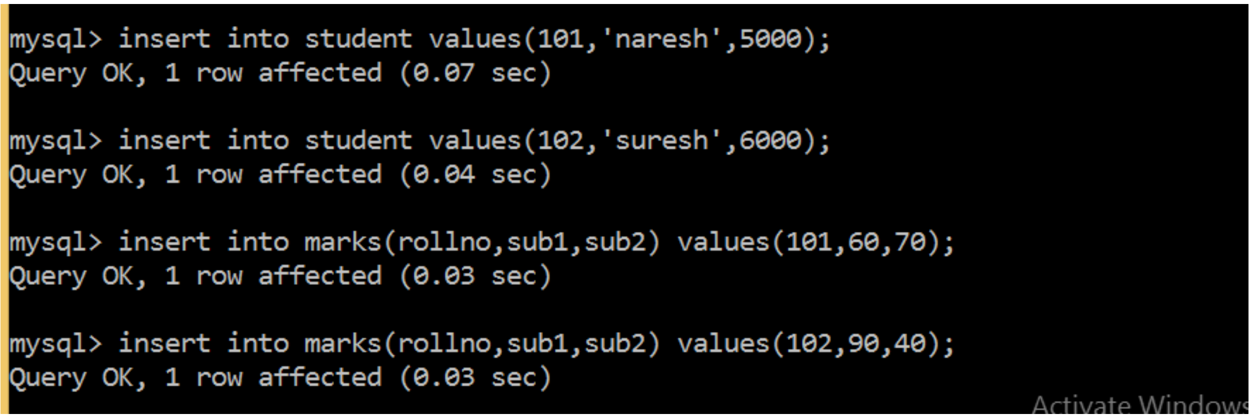
1. Insert
2. Update
3. Delete

Insert command

This command is used to insert data within table

Syntax1: insert into <table-name> values (value1,value2,…)
Syntax2: insert into <table-name>(col-name,col-name,..) values(value1,value2,value3,..)

Syntax-1 is used to insert values into all columns
Syntax-2 is used to insert values into selected columns



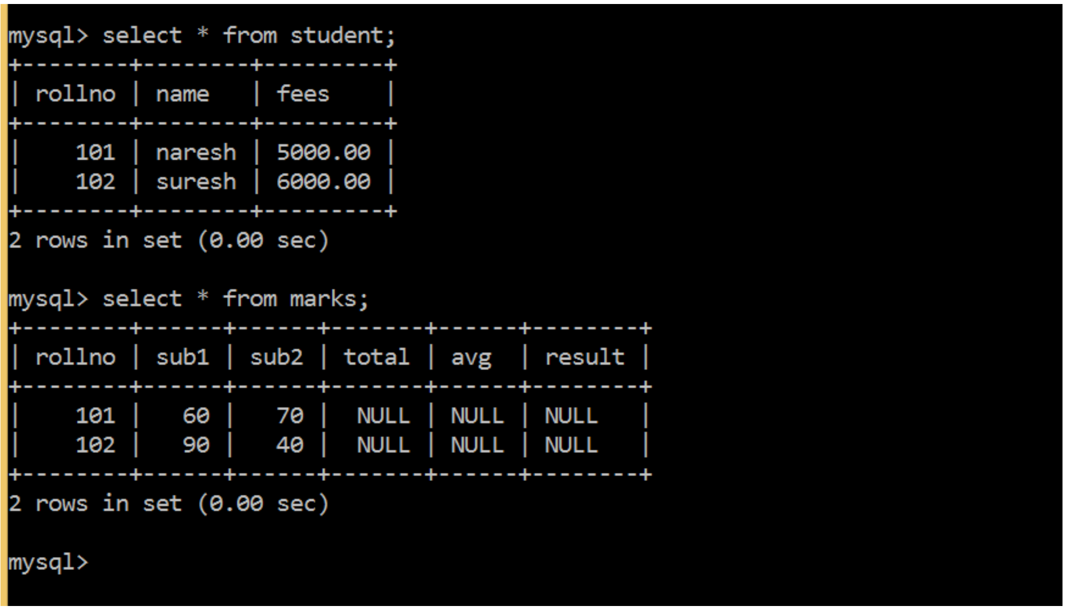
```
mysql> insert into student values(101,'naresh',5000);
Query OK, 1 row affected (0.07 sec)

mysql> insert into student values(102,'suresh',6000);
Query OK, 1 row affected (0.04 sec)

mysql> insert into marks(rollno,sub1,sub2) values(101,60,70);
Query OK, 1 row affected (0.03 sec)

mysql> insert into marks(rollno,sub1,sub2) values(102,90,40);
Query OK, 1 row affected (0.03 sec)
```

DQL/DRL (Data Query Language)

This command for read data from database table

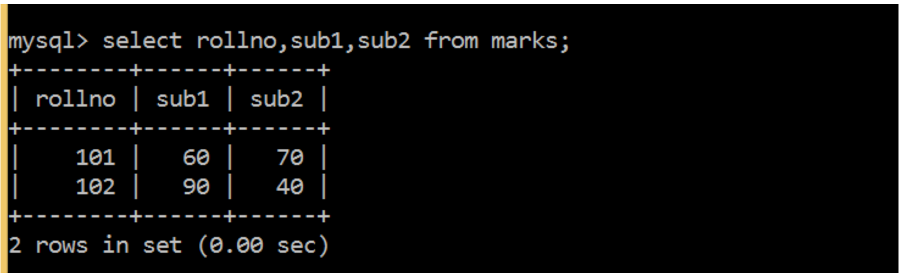
1. Select

Select * from <table-name>; → read complete table data

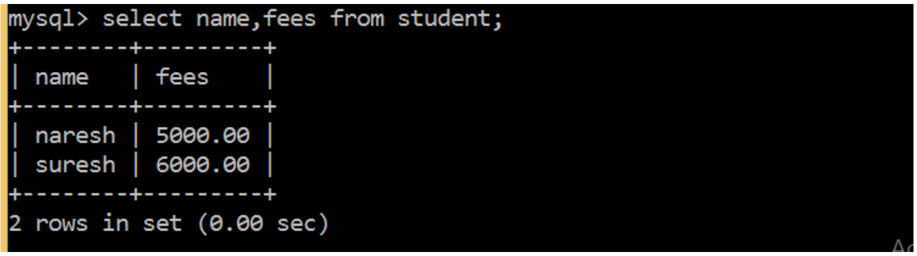


```
mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 5000.00 |
|    102 | suresh | 6000.00 |
+--------+--------+---------+
2 rows in set (0.00 sec)

mysql> select * from marks;
+--------+------+------+-------+------+--------+
| rollno | sub1 | sub2 | total | avg  | result |
+--------+------+------+-------+------+--------+
|    101 |   60 |   70 |  NULL | NULL | NULL   |
|    102 |   90 |   40 |  NULL | NULL | NULL   |
+--------+------+------+-------+------+--------+
2 rows in set (0.00 sec)

mysql>
```

Select <col-name>,<col-name>,.. from table-name;
This syntax read data from selected columns



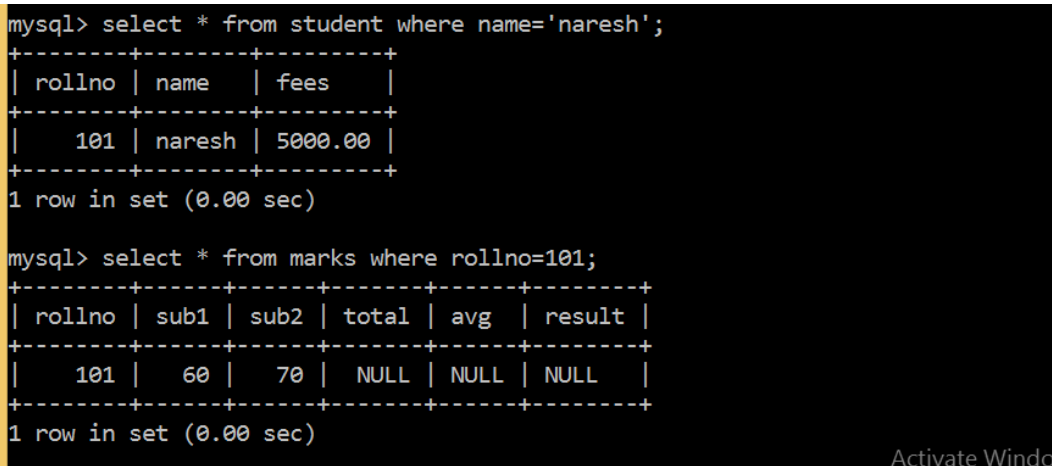
```
mysql> select rollno,sub1,sub2 from marks;
+--------+------+------+
| rollno | sub1 | sub2 |
+--------+------+------+
|    101 |   60 |   70 |
|    102 |   90 |   40 |
+--------+------+------+
2 rows in set (0.00 sec)
```



```
mysql> select name,fees from student;
+--------+---------+
| name   | fees    |
+--------+---------+
| naresh | 5000.00 |
| suresh | 6000.00 |
+--------+---------+
2 rows in set (0.00 sec)
```

Select <column –name>,<column-name>/* from <table-name> where <condition>

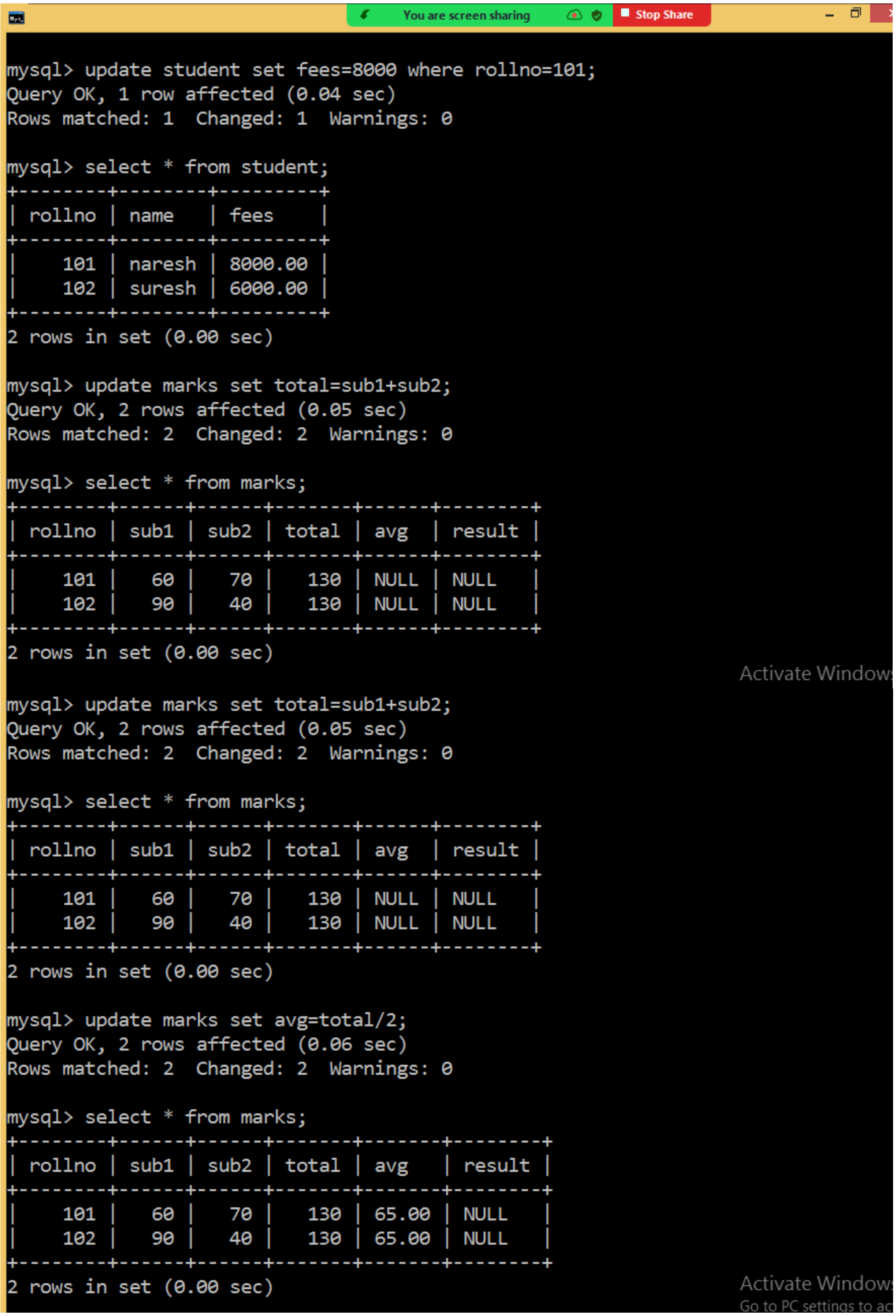
This syntax is used to select rows based condition



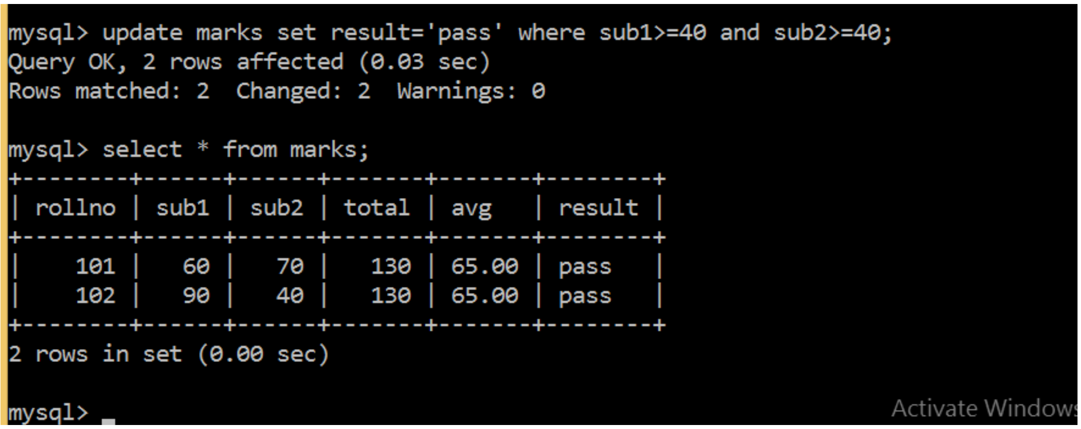
```
mysql> select * from student where name='naresh';
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 5000.00 |
+--------+--------+---------+
1 row in set (0.00 sec)

mysql> select * from marks where rollno=101;
+--------+------+------+-------+------+--------+
| rollno | sub1 | sub2 | total | avg  | result |
+--------+------+------+-------+------+--------+
|    101 |   60 |   70 |  NULL | NULL | NULL   |
+--------+------+------+-------+------+--------+
1 row in set (0.00 sec)
```

Update command
Replacing values of one or more than one column.

Update <table-name> set <column-name>=value,<column-name>=value where condition;



```
mysql> update student set fees=8000 where rollno=101;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 8000.00 |
|    102 | suresh | 6000.00 |
+--------+--------+---------+
2 rows in set (0.00 sec)

mysql> update marks set total=sub1+sub2;
Query OK, 2 rows affected (0.05 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from marks;
+--------+------+------+-------+------+--------+
| rollno | sub1 | sub2 | total | avg  | result |
+--------+------+------+-------+------+--------+
|    101 |   60 |   70 |   130 | NULL | NULL   |
|    102 |   90 |   40 |   130 | NULL | NULL   |
+--------+------+------+-------+------+--------+
2 rows in set (0.00 sec)

mysql> update marks set total=sub1+sub2;
Query OK, 2 rows affected (0.05 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from marks;
+--------+------+------+-------+------+--------+
| rollno | sub1 | sub2 | total | avg  | result |
+--------+------+------+-------+------+--------+
|    101 |   60 |   70 |   130 | NULL | NULL   |
|    102 |   90 |   40 |   130 | NULL | NULL   |
+--------+------+------+-------+------+--------+
2 rows in set (0.00 sec)

mysql> update marks set avg=total/2;
Query OK, 2 rows affected (0.06 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from marks;
+--------+------+------+-------+-------+--------+
| rollno | sub1 | sub2 | total | avg   | result |
+--------+------+------+-------+-------+--------+
|    101 |   60 |   70 |   130 | 65.00 | NULL   |
|    102 |   90 |   40 |   130 | 65.00 | NULL   |
+--------+------+------+-------+-------+--------+
2 rows in set (0.00 sec)
```

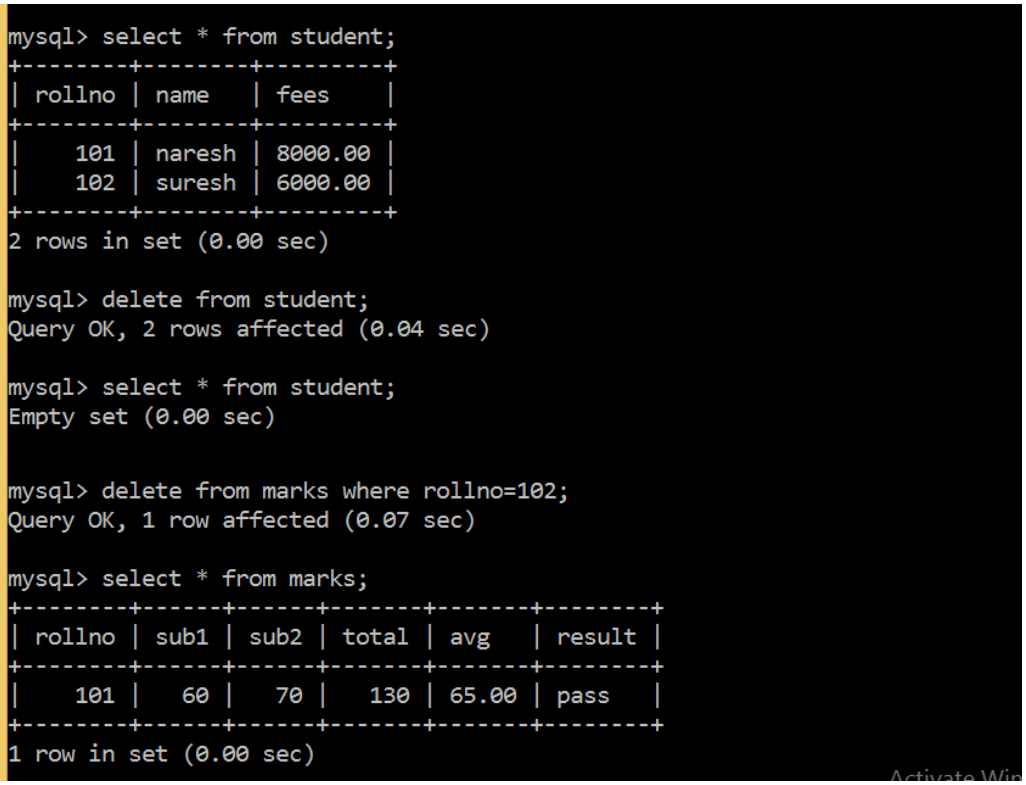


```
mysql> update marks set result='pass' where sub1>=40 and sub2>=40;
Query OK, 2 rows affected (0.03 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from marks;
+--------+------+------+-------+-------+--------+
| rollno | sub1 | sub2 | total | avg   | result |
+--------+------+------+-------+-------+--------+
|    101 |   60 |   70 |   130 | 65.00 | pass   |
|    102 |   90 |   40 |   130 | 65.00 | pass   |
+--------+------+------+-------+-------+--------+
2 rows in set (0.00 sec)

mysql> _
```

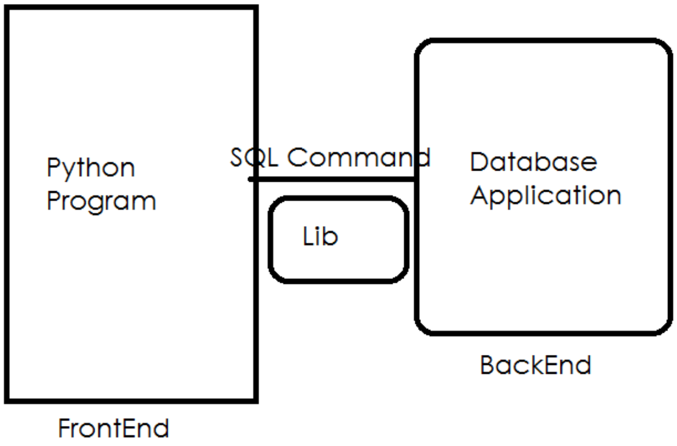
Delete command

This command is used to delete rows from database table

Delete from <table-name> where <condition>

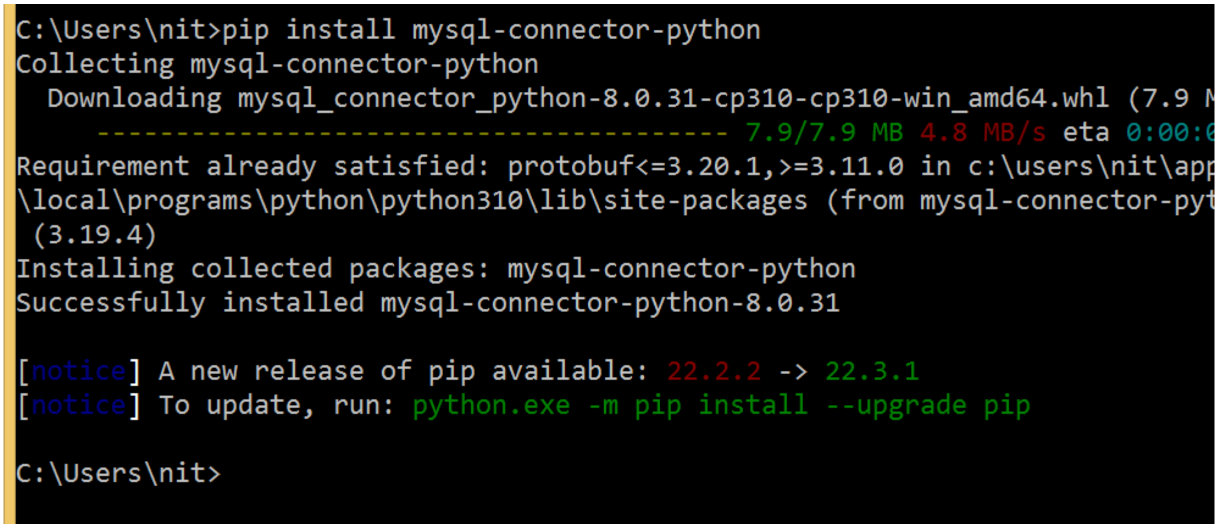


```
mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 8000.00 |
|    102 | suresh | 6000.00 |
+--------+--------+---------+
2 rows in set (0.00 sec)

mysql> delete from student;
Query OK, 2 rows affected (0.04 sec)

mysql> select * from student;
Empty set (0.00 sec)

mysql> delete from marks where rollno=102;
Query OK, 1 row affected (0.07 sec)

mysql> select * from marks;
+--------+------+------+-------+-------+--------+
| rollno | sub1 | sub2 | total | avg   | result |
+--------+------+------+-------+-------+--------+
|    101 |   60 |   70 |   130 | 65.00 | pass   |
+--------+------+------+-------+-------+--------+
1 row in set (0.00 sec)
```

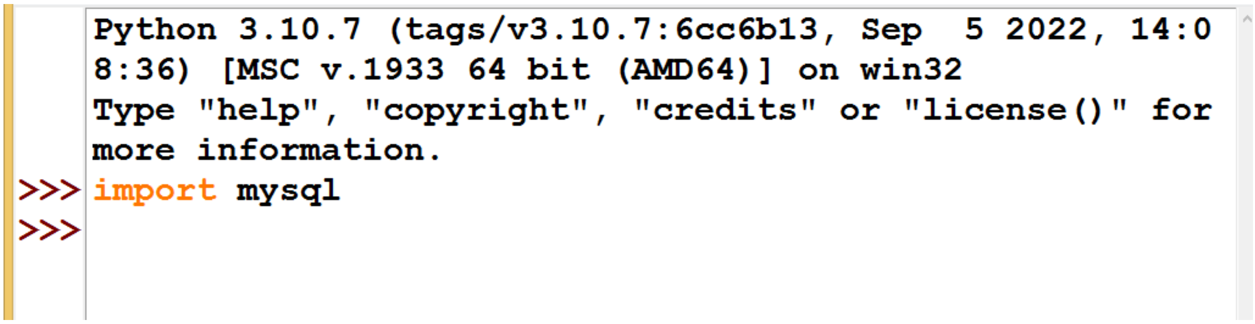


Database Vendor provides library to communicate with database software. In order to communicate with mysql database, oracle people provided one library called,

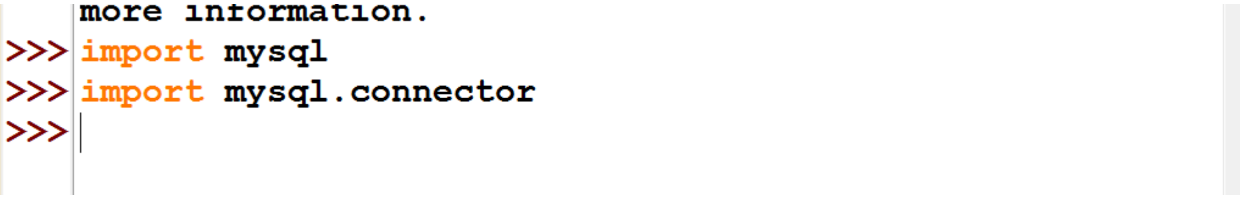
mysql-connector-python



```
C:\Users\nit>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.31-cp310-cp310-win_amd64.whl (7.9 M
     ---------------------------------------- 7.9/7.9 MB 4.8 MB/s eta 0:00:0
Requirement already satisfied: protobuf<=3.20.1,>=3.11.0 in c:\users\nit\app
\local\programs\python\python310\lib\site-packages (from mysql-connector-pyt
 (3.19.4)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-8.0.31

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\nit>
```



```
    Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:0
    8:36) [MSC v.1933 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for
    more information.
>>> import mysql
>>>
```



```
more information.
>>> import mysql
>>> import mysql.connector
>>> 
```

Basic steps to communicate with database

1. **Establish connection to database**
2. **Create cursor object**
3. **Using cursor object Send SQL statement to database**
4. **Read result stored inside cursor object**
5. **Close connection**

Establishing connection to database

mysql.connector.connect()
This function establish connection to mysql database and return connection object (MySQLConnection)

Syntax: mysql.connector.connect(database,user,password)

Example:

```
# write a python program to establish connection
# to mysql database

import mysql.connector
def main():
cn=mysql.connector.connect(database="db6pm",user="root",password="root")
    print("Connection Established...")
    print(type(cn))
main()
```

Output:

```
========= RESTART: F:/python6pmaug/dbtest1.py ========
Connection Established...
<class 'mysql.connector.connection_cext.CMySQLConnection'>
```

Cursor object

SQL statements are executed and result of SQL statement is stored inside cursor object.

How to create cursor object?
Connection provides the method called cursor(), which return MySQLCursor object.

Example:

```
# write a python program to establish connection
# to mysql database

import mysql.connector
def main():
cn=mysql.connector.connect(database="db6pm",user="root",password="ro
ot")
    print("Connection Established...")
    c=cn.cursor()
    print("Cursor is created")
    print(type(c))
main()
```

Output:

```
========= RESTART: F:/python6pmaug/dbtest1.py ========
Connection Established...
Cursor is created
<class 'mysql.connector.cursor_cext.CMySQLCursor'>
```

Cursor object provide the following methods for sending SQL statements

1. execute()
2. executeMany()

execute()
this method is used to send SQL statement to database.

Syntax: execute(SQL,params=None)

SQL statement without parameters/replacement fields is called static SQL statement

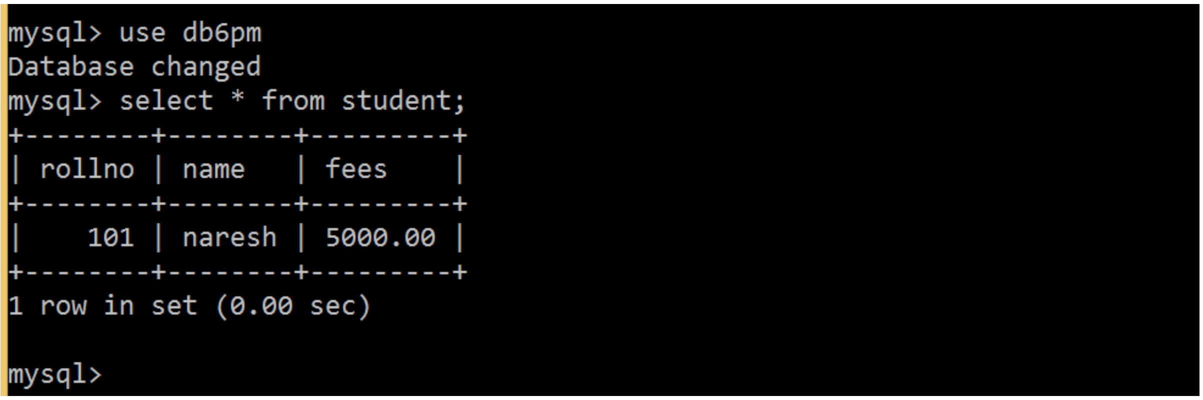
SQL statement with parameters/replacement fields is called dynamic SQL statement

Example:

```
# write a python program insert data into student table

import mysql.connector as mysql
def main():
    cn=mysql.connect(database="db6pm",user="root",password="root")
    print("connection established...")
    c=cn.cursor()
    c.execute("insert into student values(101,'naresh',5000)")
    print(f'{c.rowcount} rows are inserted')
    cn.commit()
    cn.close()

main()
```

Output:

```
========= RESTART: F:/python6pmaug/dbtest2.py ========
connection established...
1 rows are inserted
```

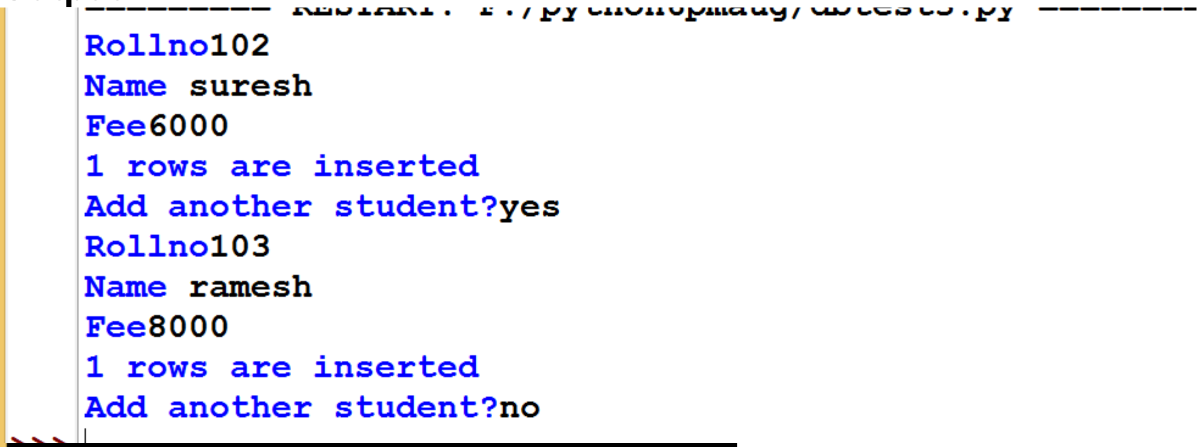


```
mysql> use db6pm
Database changed
mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 5000.00 |
+--------+--------+---------+
1 row in set (0.00 sec)

mysql>
```

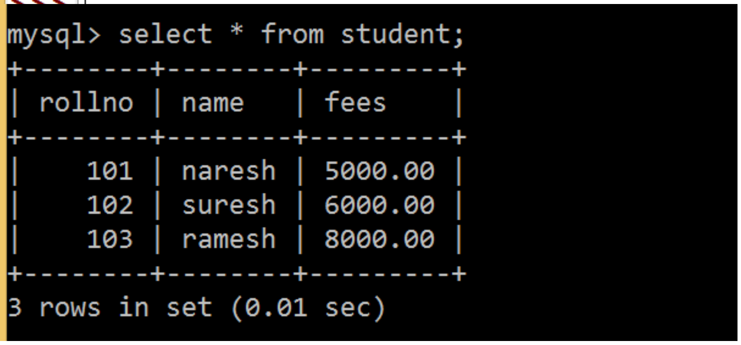
Example:

```
# program for dynamic SQL statement

import mysql.connector as mysql
def main():
    cn=mysql.connect(database="db6pm",user="root",password="root")
    c=cn.cursor()
    while True:
        r=int(input("Rollno"))
        n=input("Name ")
        f=float(input("Fee"))
        c.execute("insert into student values(%s,%s,%s)",params=[r,n,f])
        print(f'{c.rowcount} rows are inserted')
        ans=input("Add another student?")
        if ans=="no":
            cn.commit()
            cn.close()
            break
main()
```

Output:

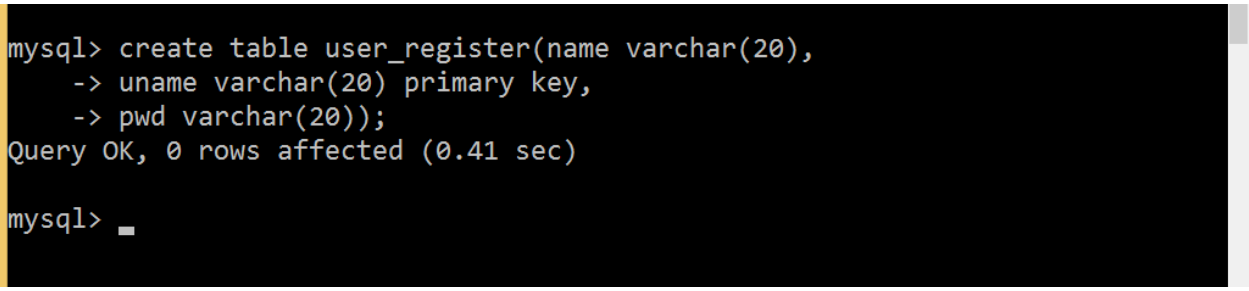


```
========== RESTART: F:/pythonpmaug/dbtest3.py ========
Rollno102
Name suresh
Fee6000
1 rows are inserted
Add another student?yes
Rollno103
Name ramesh
Fee8000
1 rows are inserted
Add another student?no
>>>
```



```
mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 5000.00 |
|    102 | suresh | 6000.00 |
|    103 | ramesh | 8000.00 |
+--------+--------+---------+
3 rows in set (0.01 sec)
```

Create the following table in mysql database



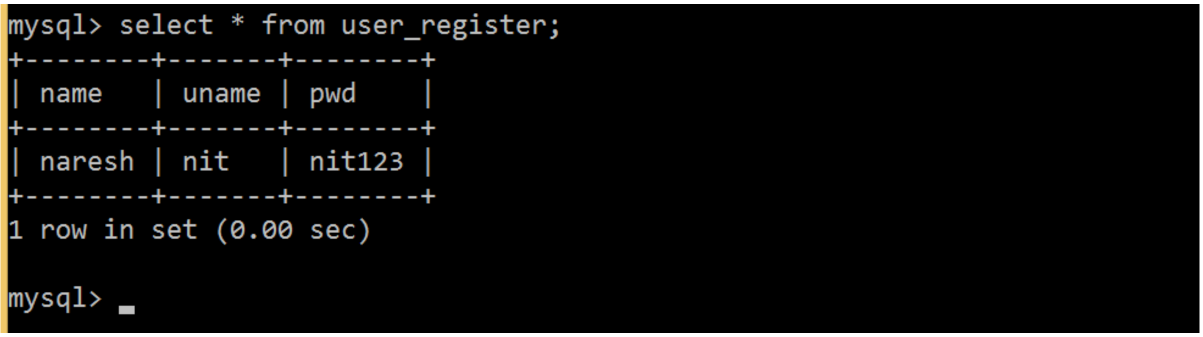
```
mysql> create table user_register(name varchar(20),
    -> uname varchar(20) primary key,
    -> pwd varchar(20));
Query OK, 0 rows affected (0.41 sec)

mysql> _
```

Example:

```
# write a program to register user

import mysql.connector

cn=mysql.connector.connect(database="db6pm",user="root",password="root")
def signup(n,u,p):
    c=cn.cursor()
    c.execute("insert into user_register values(%s,%s,%s)",params=[n,u,p])
    print("User Registered...")
    cn.commit()


def main():
    name=input("Name :")
    uname=input("UserName:")
    pwd=input("Password :")
    signup(name,uname,pwd)

main()
```

Output:

```
========= RESTART: F:/python6pmaug/dbtest4.py ========
Name :naresh
UserName:nit
Password :nit123
User Registered...
```



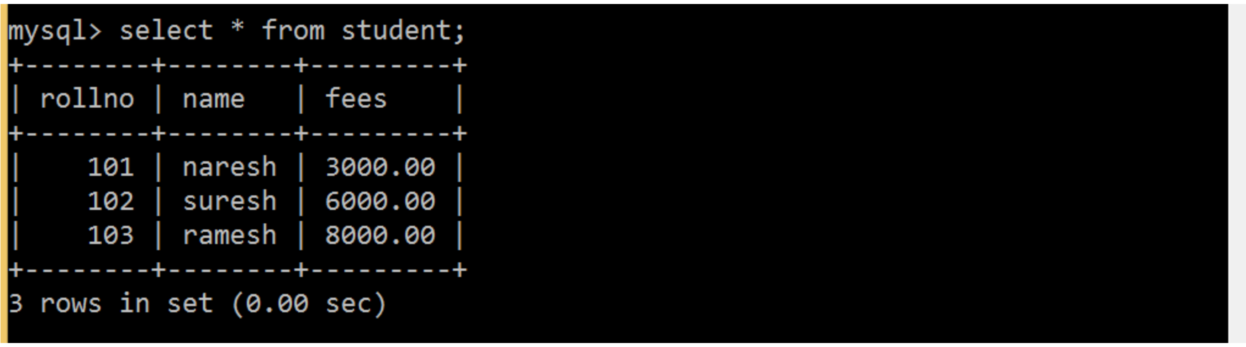
```
mysql> select * from user_register;
+--------+-------+--------+
| name   | uname | pwd    |
+--------+-------+--------+
| naresh | nit   | nit123 |
+--------+-------+--------+
1 row in set (0.00 sec)

mysql> _
```

Example:
# write a program to update fees of input rollno

```
import mysql.connector as mysql
def main():
    cn=mysql.connect(database="db6pm",user="root",password="root")
    c=cn.cursor()
    r=int(input("Enter Rollno"))
    f=float(input("Enter Fees"))
    c.execute("update student set fees=%s where rollno=%s",params=[f,r])
    k=c.rowcount
    if k>0:
        print("Updated...")
        cn.commit()
    else:
        print("Invalid Rollno")
    cn.close()
main()
```

Output:

```
========= RESTART: F:/python6pmaug/dbtest5.py ========
Enter Rollno101
Enter Fees3000
Updated...

========= RESTART: F:/python6pmaug/dbtest5.py ========
Enter Rollno105
Enter Fees6000
Invalid Rollno
```



```
mysql> select * from student;
+--------+--------+---------+
| rollno | name   | fees    |
+--------+--------+---------+
|    101 | naresh | 3000.00 |
|    102 | suresh | 6000.00 |
|    103 | ramesh | 8000.00 |
+--------+--------+---------+
3 rows in set (0.00 sec)
```