Unit 5

Connecting With Database

❖Verifying the MySQL db Interface Installation:

If the MySQLdb interface for Python has been installed successfully, we can see a new module by the name 'MySQLdb' is added to the existing modules in Python library. Go to Python IDLE shell window and type the following at Python prompt:

```
>>>help('modules')
```

As a result, it will display all the available modules of Python. We should be able to locate 'MySQLdb' module (in the beginning of the list) among them. This represents that the MySQLdb interface for Python has been successfully installed.

❖Working with MySQL Database:

We will first understand how to enter into MySQL database and issue some SQL commands to store and retrieve data. In windows 10, to open MySQL, we can use the following steps from Start button on the desktop:

```
Start→MySQL→MySQL 5.7 Command Line Client
```

It will ask for password and we have to provide the same password that we have already typed at the time of installation of MySQL.

```
Enter password: ksc123
```

Now, we can see mysql prompt where we can enter SQL commands. For example, to see which databse are existing in MySQL, we can give the following command:

```
mysql> show databases;
```

Database
information_schema
mysql
performance_schema
sakila
sys
world

The preceding output indicates that there are 6 databases in MySQL. The names of these database are displayed in the table. To know which tables are available in 'world' database, we should first enter into that database using the following command:

```
mysql> use world;
Database changed
```

To see the tables of world database, we can use the following command:

```
mysql> show tables;
```

Tables_in_world
city
country
countrylanguage

Hence, there are 3 tables in the 'world' database. Let's create a new table in the 'world' database by using 'create table'. We want to create a

table by the name ‘emptab’ with columns employee number (eno), employee name (ename) and salary (sal).

```
mysql> create table  emptab(enoint, ename char(20), sal float);
```

Let’s see the description of ‘emptab’ by giving desc command.

```
mysql>descemptab;
```

Field	Type	Null	Key	Default	Extra
eno	Int(11)	YES		NULL	
ename	char(20)	YES		NULL	
sal	float	YES		NULL	

We did not store any data into ‘emptab’.Now let’s store some rows into this table.

```
mysql> insert into emptab values(1001,”saurav”,7800);
```

```
mysql>insert into emptab values (1002,”Jay”,8900.50);
```

In this way, we can store some rows into ‘emptab’. To see all the rows of the table, we can use ‘select statement’ as:

```
mysql> select * from emptab;
```

eno	Ename	sal
1001	Saurav	7800
1002	Jay	8900.5
1003	Bhavin	9099.99
1004	Shubham	6799.0

Suppose, we want to update the salary of an employee, whose employee number is 1003,

```
mysql>update emptab set sal = sal+1000 where eno = 1003;
```

If we want to delete the row of an employee whose name is “Jay”;

Finally, to close MySQL, we can give ‘quit’ command. This will lead us to system prompt.

```
mysql> quit;
Bye
```

❖Using MySQL From Python:

To work with MySQL in a Pyhton program, we have to use MySQLdb module. We can import this module as:

```
Import MYSQLdb
```

To establish connection with MySQL database , we use the connect() method of MySQLdb module as:

```
conn = MySQLdb.connect(host=’localhost’, database=’world’, user=’root’, password=’ksc123’)
```

The connect() method returns MySQLconnection class object ’conn’. This method takes the following parameters:

host= In the network, server’ ip address should be given as host. If we use individual computer, then the string ‘localhost’ can be supplied.

database= This is the database name to connect to.

user= This represents the user name. Generally this is ‘root’.

password= This indicates the root password.

The next step is to create cursor class object by calling the cursor() method on ‘conn’ object as:

```
cursor = conn.cursor()
```

The ‘cursor’ object is useful to execute SQL commands on the database. For this purpose, we can use the execute() method of ‘cursor’ object as:

```
cursor.execute("select * from emptab")
```

The resultant rows retrieved from the table are stored in the 'cursor'. We can get them from the 'cursor' object using the fetchone() or fetchall() methods.

```
row = cursor.fetchone()
```

```
rows=cursor.fetchall()
```

Finally, we can close the connection with MySQL by closing the cursor and connection object as:

```
cursor.close()
```

```
conn.close()
```

❖Retrieving All Rows From Table:

After establishing connection with MySQL database using connect() method, we have to create 'cursor' object. Then we have to call the execute() method which executes our SQL command as:

```
cursor.execute("select  * from emptab")
```

Our assumption in this case is that we have 'emptab' table already available in the 'world' database in MySQL. The rows of the 'emptab' will be available in the 'cursor' object. We can retrieve them by using the fetchone() method as shown below:

```
row = cursor.fetchone() #retrieve the first row
while row is not None:   #if that row exists then
        print(row)           #display it
        row = cursor.fetchone()   #get the next row
```

This logic is shown in the program where we are retrieving all the rows of the 'emptab' from MySQl database software.

Program 1: A Python program to retrieve and display all rows from the employee table.

```
#displaying all rows of emptab
import MySQLdb
#connect to mysql database
conn = MySQLdb.connect(host='localhost', database='world', user='root',
password='ksc123')
#prepare a cursor object using cursor() method
cursor=conn.cursor()
#execute a sql query using execute() method
cursor.execute("select * from emptab")
#get only one row
row=cursor.fetchone()
#if the row exists
while row is not None:
    print(row)
    row=cursor.fetchone() #get the next row
#close connection
cursor.close()
conn.close()
```

We can also use the fetchall() method that retrieves all rows from the 'emptab' table and stores them in an iterator, as:

```
rows = cursor.fetchall()
```

Now, we can retrieve the rows one by one from 'rows' iterator using a for loop as:

```
for row in rows:
    print(row)
```

Program 2: A Python program to retrieve and display all rows from the employee table.

```
import MySQLdb
# connect to MySQL database
conn = MySQLdb.connect(host='localhost', database='world', user='root',
password='ksc123')
# prepare a cursor object using cursor() method
cursor = conn.cursor()
# execute a SQL query using execute() method
cursor.execute("select * from emptab ")
# get all rows
rows = cursor.fetchall()
# display the number of rows
print('Total number of rows= ', cursor.rowcount)
# display the rows from rows object
for row in rows :
 print (row)  # display each row
# close connection
```

```
cursor.close()
conn.close()
```

Please observe the output of the previous programs. The output is displayed in the form of tuples. We can format the output and display the rows in better manner. Since we know that the rows are available in the 'rows' iterator, we can use a for loop and retrieve eno, ename and sal values which are the columns in the table as:

for row in rows:

eno = row[0] # first column value

ename = row[1] # second column value

sal = row [2] # third column value

print('%-60%-15s %10.2f'% (eno, ename, sal))

This is shown in Program 3 which is a slight improvement of Program 2.

Program 3: A Python program to retrieve all rows from employee table and display the column values in tabular form.

❖
```
# displaying all rows of emptab V3.0
import MySQLdb
# connect to MySQL database
conn = MySQLdb.connect(host='localhost', database='world', user='root',
password='ksc123')
# prepare a cursor object using cursor() method
cursor = conn.cursor()
# execute a SQL query using execute() method
cursor.execute("select * from emptab")
# get all rows
rows = cursor.fetchall()
# display the number of rows
print('Total number of rows= ', cursor.rowcount)
# display the rows from rows object
for row in rows :
 eno = row[0]
 ename = row[1]
 sal = row[2]
 print('%-6d %-15s %10.2f'% (eno, ename, sal))
# close connection
cursor.close()
conn.close()
```

❖Inserting Rows Into a Table:

To insert a row into a table, we can use SQL insert command as:

```
str = "insert into emptab(eno, ename, sal) values(9999, 'Srinivas', 9999.99)"
```

Here, we are inserting values for eno, ename and sal columns into 'emptab' table. The total command is taken as a string 'str'. Now we can execute this command by passing it to execute() method as:

```
cursor.execute(str) # SQL insert statement in the str is executed
```

After inserting a row into the table, we can save it by calling the commit() method as:

```
conn.commit()
```

In case there is an error, we can un-save the row by calling the rollback() method as:

```
conn.rollback()
```

Program 4: A Python program to insert a row into a table in MySQL.

```
# inserting a row into emptab
import MySQLdb
# connect to MySQL database
conn = MySQLdb.connect(host='localhost', database='world', user='root',
password='ksc123')
# prepare a cursor object using cursor() method
cursor = conn.cursor()
# prepare SQL query string to insert a row
str = "insert into emptab(eno, ename, sal) values (9999, 'Srinivas', 9999.99)"
try:
# execute the SQL query using execute() method
 cursor.execute(str)
# save the changes to the database
 conn.commit()
 print('1 row inserted...')
except:
# rollback if there is any error
 conn.rollback()
# close connection
cursor.close()
conn.close()
```

We can improve this program so that it is possible to insert several rows into emptab from the keyboard. For this purpose, we can write an insert_rows() function as:

```
def insert_rows (eno, ename, sal):
```

When this function is called, we have to pass values for the arguments eno, ename and sal. These values can be stored in a tuple as:

```
args = (eno, ename, sal)
```

Now, our SQL command for inserting a row can be:

```
str = "insert into emptab(eno, ename, sal) values ('%d', '%s', '%f')"
```

Here, we are just mentioning the '%d', '%s' and '%f" as format strings for values. These format strings assume the values from ‘args' supplied to the function. Now, we can call the execute() method as:

```
cursor.execute(str % args)
```

Please observe the '%' symbol between the SQL command string 'str' and the args' tuple. This is a better way of using execute() method. The string 'str' will be executed by substituting the argument values in the place of format strings. If we pass 10, Venu' and 7777.77 as values for the arguments, the string will become:

```
insert into emptab(eno, ename, sal) values (10, Venu', 7777.77)
```

To understand this simple substitution, consider Figure :

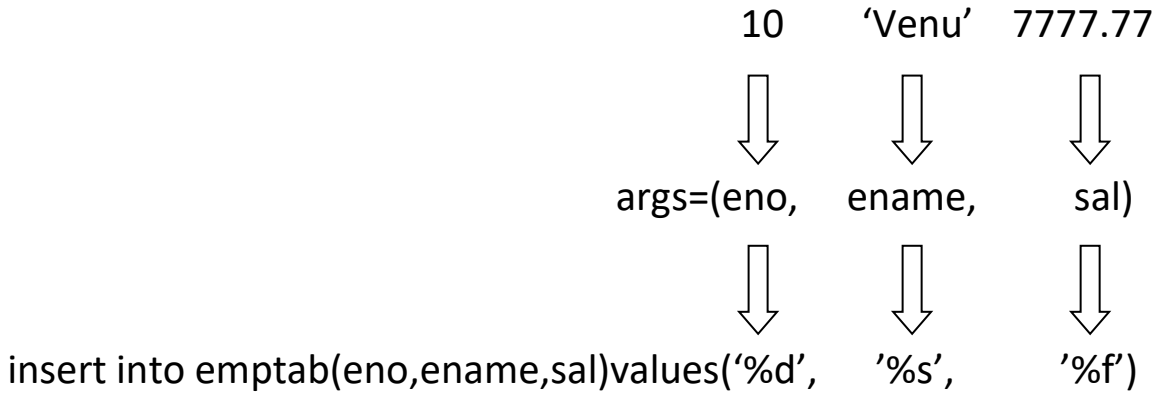

Figure:Argument Values are Passed to SQL Command

Program 5: A Python program to insert several rows into a table from the keyboard.

```
# inserting several rows from keyboard into emptab table
import MySQLdb
 # function to store row into the emptab table
def insert_rows (eno, ename, sal):
 # connect to MySQL database
  conn=MySQLdb.connect(host='localhost', database='world', user='root',
password='ksc123')
 # prepare a cursor object using cursor() method
  cursor = conn.cursor()
 # prepare SQL query string to insert a row
  str = "insert into emptab(eno, ename, sal) values('%d', '%s','%f')"
 # define the arguments
  args=(eno,ename,sal)
  try:
   # execute the SQL query using execute() method
   cursor.execute(str % args)
   # save the changes to the database
   conn.commit()
   print('1 row inserted...')
  except:
   # rollback if there is any error
   conn.rollback()
  finally:
   # close connection
   cursor.close()
   conn.close()
   # enter rows from keyboard
n = int(input('How many rows?'))
for i in range(n):
 X=int(input('Enter eno: '))
 Y = input('Enter name: ')
 Z=float(input('Enter salary:'))
 # pass eno, name and salary to insert_rows() function
 insert_rows(X,Y,Z)
 print(' --------------------')
```

❖Deleting Rows From a Table:

We can accept the employee number from the keyboard and delete the corresponding row in the employee table. For this purpose, we can use the SQL command as:

```
delete from emptab where eno = '%d'
```

Here, '%d' represents the integer number supplied to the command as argument. This is nothing but the employee number, i.e., eno.

Program 6: A Python program to delete a row from emptab by accepting the employee number.

```
#deleting a row from emptab depending on eno
import MySQLdb
# function to delete row from the emptab table
def delete_rows (eno):
 conn = MySQLdb.connect(host='localhost',  database='world', user='root',
password='ksc123')
 cursor = conn.cursor()
 str ="delete from emptab where eno = '%d'"
 args = (id)
 try:
  cursor.execute(str % args)
  conn.commit()
  print('1 row deleted...')
 except:
  conn.rollback()
 finally:
  cursor.close()
  conn.close()
X =int(input('Enter eno: '))
# pass eno to delete_rows() function
delete_rows (X)
```

❖Updating Rows In a Table:

To modify the values in a row, we can use update command. For example, to increase the salary of an employee by 1000, we can write:

```
update emptab set sal sal+1000 where eno = '%d'
```

Here, '%d' represents the argument value, i.e. the employee number. This can be passed from the keyboard

Program 7: A Python program to increase the salary of an employee by employee number from keyboard.

```
# updating the salary in emptab depending on eno
from MySQLdb import *
# function to update row from the emptab table
def update_rows(eno):
    conn = connect(host='localhost', database='world', user='root', password='ksc123')
    cursor=conn.cursor()
    str="update emptab set sal = sal+1000 where id = '%d'"
    args = (id)
    try:
        cursor.execute(str % args)
        conn.commit()
        print('1 row updated...')
    except:
        conn.rollback()
    finally:
        cursor.close()
```

```
        conn.close()
X=int(input('Enter eno: '))
update_rows(X)
```

❖Creating Database Tables Through Python:

So far, we used ‘emptab' table that was already created in the MySQL database. It is also possible to create any table in the MySQL database using a Python program. Once the table is created, we can insert the rows through a Python program. Suppose we want to create emptab table with the columns: eno, ename, sex and salary, we can write a SQL command as follows:

```
create table emptab(enoint, ename char(20), sex char(1), salary float)
```

We can understand that eno is integer type, ename is declared as 20 characters sixe, sex as 1 character size and salary as a float type column.

Before the emptab is created, we can check if any table already exists by that name in the database and we can delete that table using drop table command as:

```
drop table if exists emptab
```

The above command should be first executed and after that, we can create the new emptab table.

Program 8: A Python program to create emptab in MySQL database.

```
#creating a table by the name emptab in MySQL database
from MySQLdb import *
#connect to MySQL database
conn =connect(host='localhost', database='world', user='root', password='ksc123')
#prepare a cursor object using cursor() method
cursor = conn.cursor()
#delete table if already exists
cursor.execute("drop table if exists emptab")
#prepare sql string to create a new table
str = "create table emptab(eno int, ename char(20), sex char(1), salary float)"
#execute the query to create the table
cursor.execute(str)
print("Table Created.........")
#close connection
cursor.close()
conn.close()
```

To check if the emptab is properly created, we can go to MySQL prompt and give the following command:

mysql >desc emptab;

Field	Type	Null	Key	Default	Extra
eno	int(11)	YES		NULL	
ename	char(20)	YES		NULL	
sex	char(1)	YES		NULL	
sal	float	YES		NULL	

It would be better if we can use GUI environment to connect and communicate with a database. In the following program, we are displaying an Entry box where the user enters employee number and retrieves the corresponding employee row from the 'emptab' table. The row is again displayed using a lable in the Frame.

Program 9: A Python program using GUI to retrieve a row from a from MySQL database table.

```
#retrieving rows from a table in MySQL databse – a GUI application
import MySQLdb
from tkinter import*
#create root window
root = Tk()
#a function that takes employee number and displays the row
def retrieve_rows(id):
    # connect to MySQL database

conn=MySQLdb.connect(host='localhost',database='world',user='root',password='ksc123')
    # prepare a cursor object using cursor() method
    cursor = conn.cursor()
    # prepare SQL query string to retrieve a row
    str="select * from emptab where eno ='%d'"
    # define the arguments
    args = (eno)
    # execute the SQL query using execute() method
    cursor.execute(str % args)
    # get only one row
    row = cursor.fetchone()
    if row is not None:
        lbl = Label(text= row, font=('Arial', 14)).place(x=50, y=200)
    # close connection
    cursor.close()
    conn.close()
    # a function that takes input from Entry widget
def display(self):
    # retrieve the value from the entry widget
```

```
    str=e1.get()
    # display the value using label
    lbl= Label (text='You entered:' +str, font=('Arial', 14)).place(x=50, y=150)
    # call the function that retrieves the row
    retrieve_rows(int(str))
    # create a frame as child to root window
f = Frame(root, height=350, width=600)
#let the frame will not shrink
f.propagate(0)
#attach the frame to root window
f.pack()
#label
l1 = Label(text='Enter employee number: ', font=('Arial', 14))
#create entry widget for accepting employee number
e1 = Entry (f, width=15, fg='blue',bg='yellow', font=('Arial', 14))
#when user presses enter,bind that event to display method
e1.bind("<Return>", display)
#place label and entry widgets in the frame
l1.place (x=50, y=100)
e1.place(x=300, y=100)
#handle the events
root.mainloop()
```