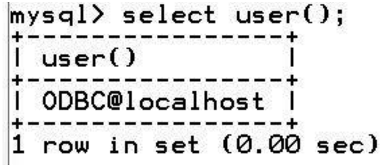
Python and MySQL Interface

In real-life scenarios, users interact with applications whereas data is stored in the database. We have already developed Python applications and database tables. Now, we will connect the Python application and MySQL database.

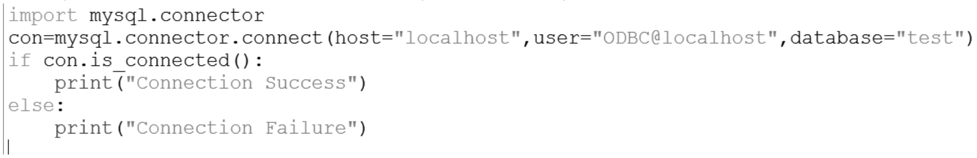
Steps for creating Python and MySQL connectivity applications:

1. **import mysql.connector**
2. **Create a connection**
   <connection object> =mysql.connector.connect (host=”localhost”, user=<username>, passwd= <password>, database =<database name>)
   Username = To know the user name, run the **select user();** command on your MySQL.



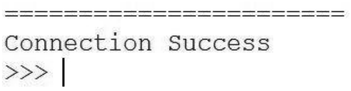
```
mysql> select user();
+-----------------+
| user()          |
+-----------------+
| ODBC@localhost  |
+-----------------+
1 row in set (0.00 sec)
```

Password = It should be the same as we set during MySQL installation.
Example: Let’s create a connection of Python with MySQL and test it.



```
import mysql.connector
con=mysql.connector.connect(host="localhost",user="ODBC@localhost",database="test")
if con.is_connected():
    print("Connection Success")
else:
    print("Connection Failure")
```

Output:



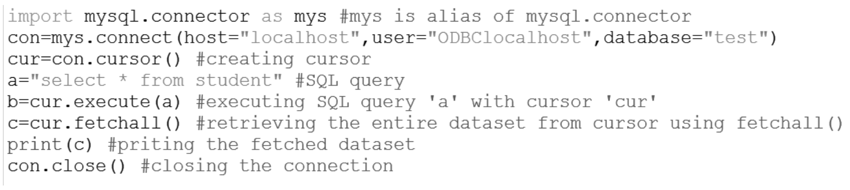
```
======================
Connection Success
>>> |
```

Here in the above example, I had not set any database password, so, the passwd option is not mentioned in the connection string. **is_connected()** function checks whether the connection string can connect Python with MySQL or not. If the connection string can make a connection with Python and MySQL, is_connected() returns True otherwise False.

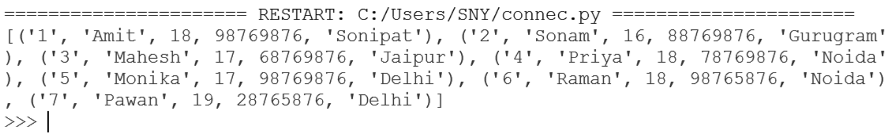
3. **Create a cursor**
   Cursor: It is a pointer or iterator which points towards the resultset of the SQL query. Whenever a SQL query runs, It gives the entire result set in one go. We may not require the entire resultset at once. So, a cursor is created and data from the entire resultset will be fetched row by row as per our requirement.
   Syntax: <cursor object> = <connection object>.cursor()

4. **Execute query**
   Syntax: <cursor object>. execute(<SQL query string>)

5. **Extract data from the result set**
   As we know, Data from the database is retrieved using the **select** query. After running the select query, we get the resultset. Now to fetch the data from the resultset, the following functions are used:
   a) **fetchall():** It returns all the records from the resultset. Each record will be in the form of a tuple whereas the entire resultset will be in the form of a list.
   Syntax: <variable name>=<cursor>.fetchall()

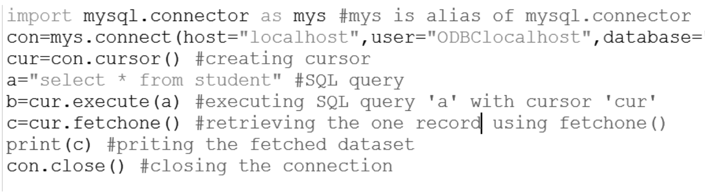


```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving the entire dataset from cursor using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

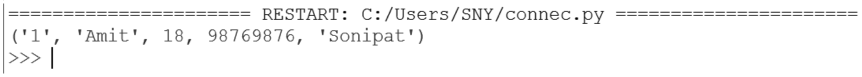


```
===================== RESTART: C:/Users/SNY/connec.py =====================
[('1', 'Amit', 18, 98769876, 'Sonipat'), ('2', 'Sonam', 16, 88769876, 'Gurugram'
), ('3', 'Mahesh', 17, 68769876, 'Jaipur'), ('4', 'Priya', 18, 78769876, 'Noida'
), ('5', 'Monika', 17, 98769876, 'Delhi'), ('6', 'Raman', 18, 98765876, 'Noida')
, ('7', 'Pawan', 19, 28765876, 'Delhi')]
>>> |
```

b) **fetchone():** It returns one row from the resultset in the form of a tuple. It returns None if no more records are there. To get multiple rows, we need to run fetchone() multiple times.
Syntax: <variable name>=<cursor>.fetchone()

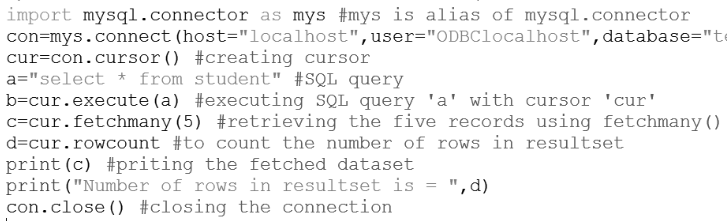


```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchone() #retrieving the one record using fetchone()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

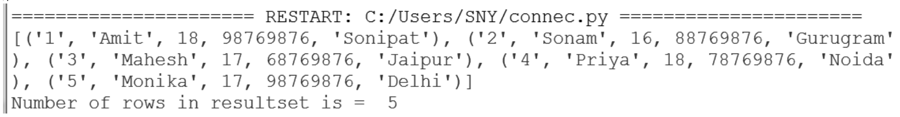


```
===================== RESTART: C:/Users/SNY/connec.py =====================
('1', 'Amit', 18, 98769876, 'Sonipat')
>>>
```

c) **fetchmany():** It returns n number of records from the resultset in the form of a list where each record is in the form of a tuple. It returns an empty tuple if no more records are there.
Syntax: <variable name>=<cursor>.fetchmany(n)



```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchmany(5) #retrieving the five records using fetchmany()
d=cur.rowcount #to count the number of rows in resultset
print(c) #priting the fetched dataset
print("Number of rows in resultset is = ",d)
con.close() #closing the connection
```



```
===================== RESTART: C:/Users/SNY/connec.py =====================
[('1', 'Amit', 18, 98769876, 'Sonipat'), ('2', 'Sonam', 16, 88769876, 'Gurugram'
), ('3', 'Mahesh', 17, 68769876, 'Jaipur'), ('4', 'Priya', 18, 78769876, 'Noida'
), ('5', 'Monika', 17, 98769876, 'Delhi')]
Number of rows in resultset is =  5
```

d) **rowcount:** It is the cursor's property to count the number of rows in the resultset.
Syntax: <variable name>=<cursor>.rowcount
Example: Refer to the example of fetchmany()

6. **Close the connection**
After doing all the processing, the connection should be closed.
Syntax: <connection object>.close()
Example: Refer to the example of fetchmany()

Format specifier: we need a format specifier to write SQL queries based on user input. To do this we have two ways:

1. **Using % formatting:**
Whenever we need to complete an SQL query based on user input, we write a placeholder '%s' on that place.
Example: Let's fetch all the data of the student table where age is 'x' and city is 'y'. (Here, the value of x and y will be given by the user during run time).

```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=int(input("Enter Student's Age = "))
y=input("Enter Student's City = ")
a="select * from student where Age=%s and Address = '%s'"%(x,y)
#SQL Query using % formatting where age and address is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving all records using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

```
===================== RESTART: C:/Users/SNY/connec.py =====================
Enter Student's Age = 18
Enter Student's City = Sonipat
[('1', 'Amit', 18, 98769876, 'Sonipat')]
>>>
```

2. **Using format() function:**
   Whenever we need to complete an SQL query based on user input, we write a placeholder {} on that place. If we need to complete the SQL query based on multiple user inputs, we write placeholder {}, {}, {} and so on at those places and pass user-defined values in the format function sequentially. Here 1st values passed in the format function will be passed to 1st {}, 2nd values passed to 2nd {}, 3rd value passed to 3rd {} and so on.
   Example: Let’s fetch all the data of the student table where age is ‘x’ and city is ‘y’. (Here, the value of x and y will be given by the user during run time).

```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=int(input("Enter Student's Age = "))
y=input("Enter Student's City = ")
a="select * from student where Age={} and Address='{}'".format(x,y)
#SQL Query using format() function where age and address is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving all records using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

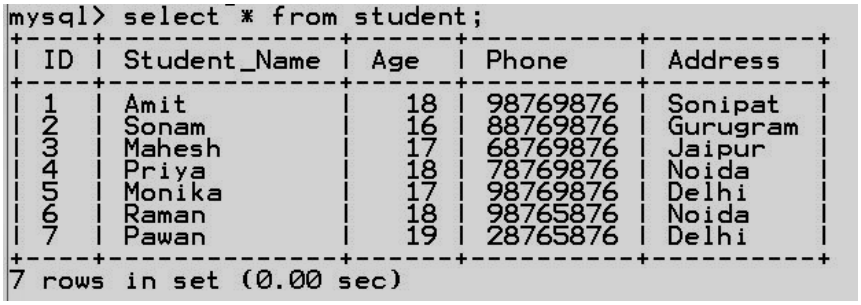
```
Enter Student's Age = 18
Enter Student's City = Sonipat
[('1', 'Amit', 18, 98769876, 'Sonipat')]
>>>
```

Commit: Whenever we perform update, delete or insert query, the commit() function must be run before closing the connection.

Syntax: <connection object>.commit()

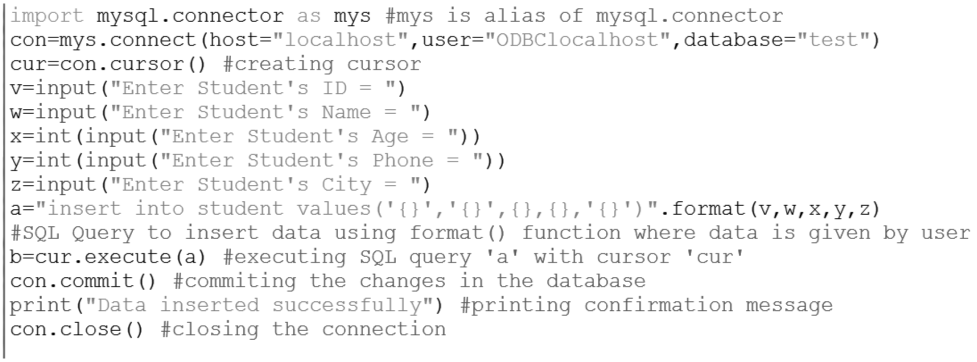
Example 1: Insert data in the student table which will be given by the user during run time.

Data in the student table before insertion



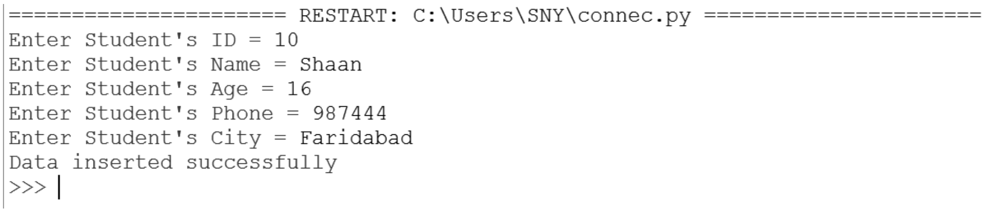
```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | Gurugram |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |
| 4  | Priya        |   18 | 78769876 | Noida    |
| 5  | Monika       |   17 | 98769876 | Delhi    |
| 6  | Raman        |   18 | 98765876 | Noida    |
| 7  | Pawan        |   19 | 28765876 | Delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

Python code:



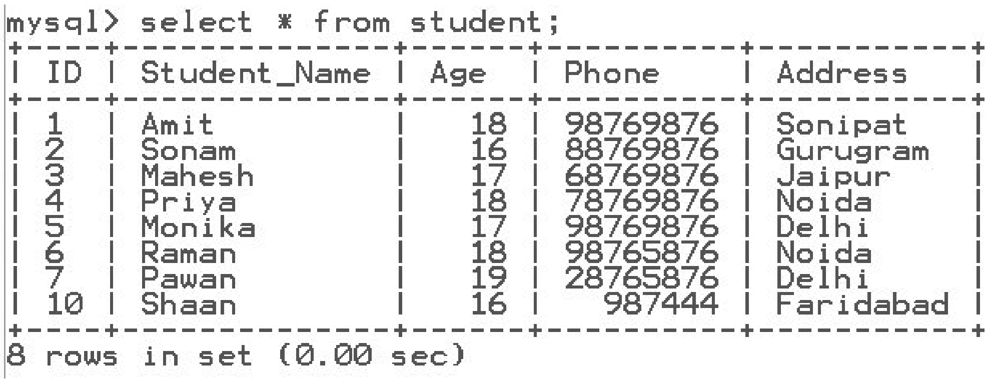
```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
v=input("Enter Student's ID = ")
w=input("Enter Student's Name = ")
x=int(input("Enter Student's Age = "))
y=int(input("Enter Student's Phone = "))
z=input("Enter Student's City = ")
a="insert into student values('{}','{}',{},{},'{}')".format(v,w,x,y,z)
#SQL Query to insert data using format() function where data is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data inserted successfully") #printing confirmation message
con.close() #closing the connection
```

Output:



```
====================== RESTART: C:\Users\SNY\connec.py ======================
Enter Student's ID = 10
Enter Student's Name = Shaan
Enter Student's Age = 16
Enter Student's Phone = 987444
Enter Student's City = Faridabad
Data inserted successfully
>>> |
```

Data in the student table after insertion

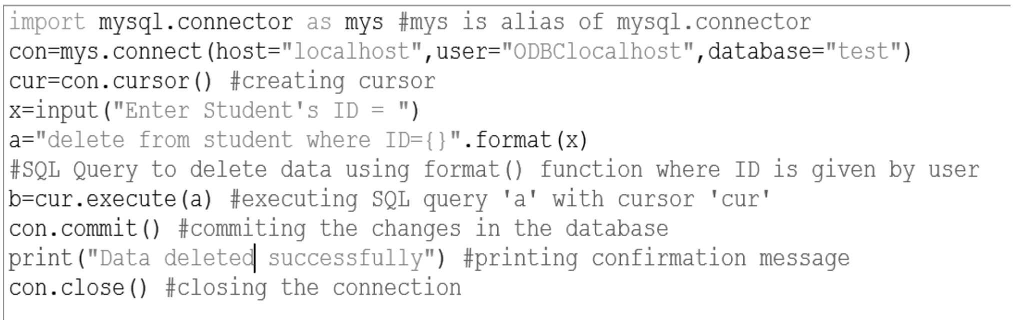


```
mysql> select * from student;
+----+--------------+------+----------+-----------+
| ID | Student_Name | Age  | Phone    | Address   |
+----+--------------+------+----------+-----------+
| 1  | Amit         |   18 | 98769876 | Sonipat   |
| 2  | Sonam        |   16 | 88769876 | Gurugram  |
| 3  | Mahesh       |   17 | 68769876 | Jaipur    |
| 4  | Priya        |   18 | 78769876 | Noida     |
| 5  | Monika       |   17 | 98769876 | Delhi     |
| 6  | Raman        |   18 | 98765876 | Noida     |
| 7  | Pawan        |   19 | 28765876 | Delhi     |
| 10 | Shaan        |   16 |   987444 | Faridabad |
+----+--------------+------+----------+-----------+
8 rows in set (0.00 sec)
```

Example 2: Delete data of those students whose ID is given by the user during run time in the student table.

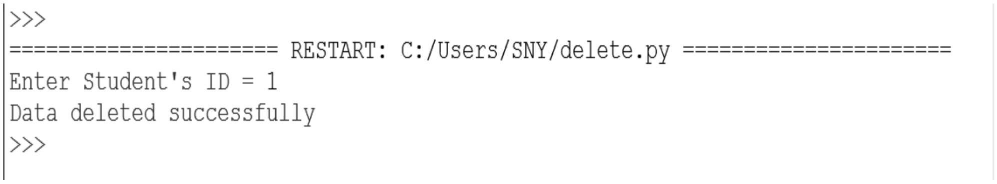
In the previous example, after inserting a record in the student table, we have 8 records in the student table. (Kindly refer previous example for the current state of the student table).

Python code:



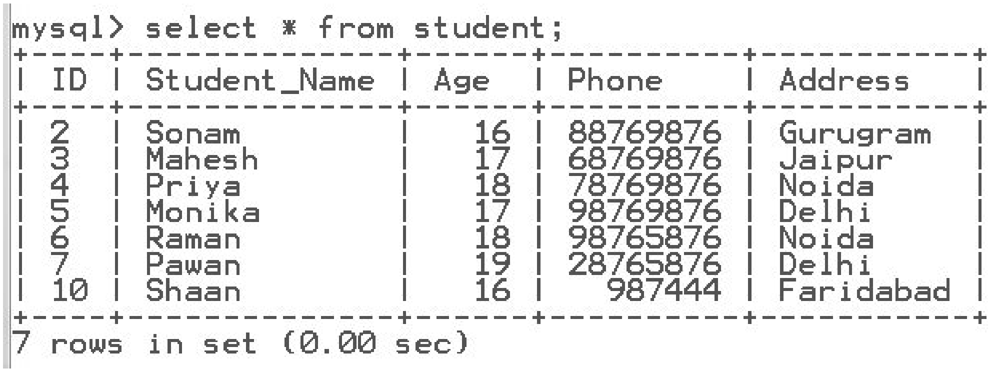
```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=input("Enter Student's ID = ")
a="delete from student where ID={}".format(x)
#SQL Query to delete data using format() function where ID is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data deleted successfully") #printing confirmation message
con.close() #closing the connection
```

Output:



```
>>>
===================== RESTART: C:/Users/SNY/delete.py =====================
Enter Student's ID = 1
Data deleted successfully
>>>
```

Data in the student table after deletion

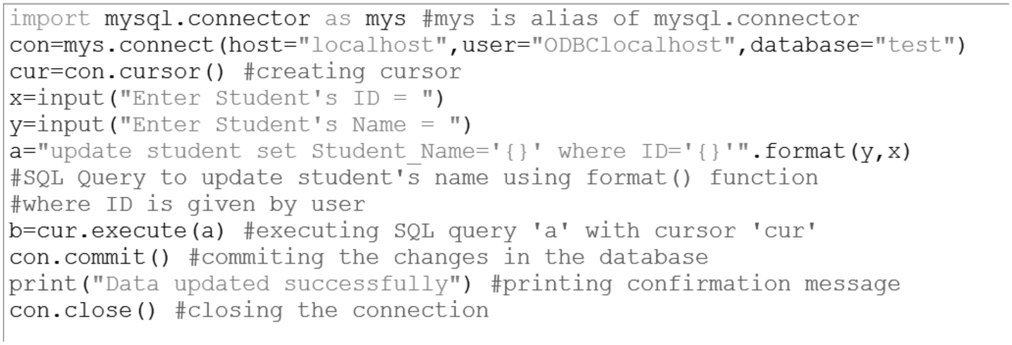


```
mysql> select * from student;
+----+---------------+------+----------+-----------+
| ID | Student_Name  | Age  | Phone    | Address   |
+----+---------------+------+----------+-----------+
| 2  | Sonam         |   16 | 88769876 | Gurugram  |
| 3  | Mahesh        |   17 | 68769876 | Jaipur    |
| 4  | Priya         |   18 | 78769876 | Noida     |
| 5  | Monika        |   17 | 98769876 | Delhi     |
| 6  | Raman         |   18 | 98765876 | Noida     |
| 7  | Pawan         |   19 | 28765876 | Delhi     |
| 10 | Shaan         |   16 |   987444 | Faridabad |
+----+---------------+------+----------+-----------+
7 rows in set (0.00 sec)
```

Example 3: Update the name of those students whose name and ID are given by the user during run time in the student table.

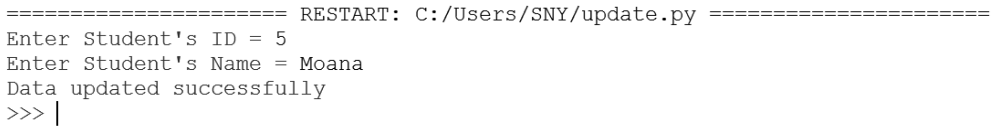
In the previous example, after deleting a record in the student table, we have 7 records in the student table. (Kindly refer previous example for the current state of the student table).

Python code:



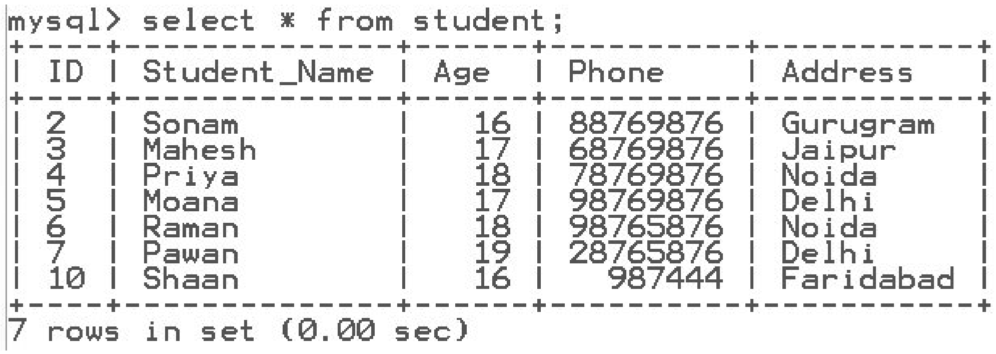
```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=input("Enter Student's ID = ")
y=input("Enter Student's Name = ")
a="update student set Student_Name='{}' where ID='{}'".format(y,x)
#SQL Query to update student's name using format() function
#where ID is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data updated successfully") #printing confirmation message
con.close() #closing the connection
```

Output:



```
====================== RESTART: C:/Users/SNY/update.py ======================
Enter Student's ID = 5
Enter Student's Name = Moana
Data updated successfully
>>> |
```

Data in student table after updation:



```
mysql> select * from student;
+----+--------------+------+----------+-----------+
| ID | Student_Name | Age  | Phone    | Address   |
+----+--------------+------+----------+-----------+
| 2  | Sonam        |   16 | 88769876 | Gurugram  |
| 3  | Mahesh       |   17 | 68769876 | Jaipur    |
| 4  | Priya        |   18 | 78769876 | Noida     |
| 5  | Moana        |   17 | 98769876 | Delhi     |
| 6  | Raman        |   18 | 98765876 | Noida     |
| 7  | Pawan        |   19 | 28765876 | Delhi     |
| 10 | Shaan        |   16 |   987444 | Faridabad |
+----+--------------+------+----------+-----------+
7 rows in set (0.00 sec)
```