Ch 9 : Interface Python with MySQL

Connecting to MySQL From Python

MySQL connector library helps to connect to a MySQL database from within a Python script.

Steps - Creating Database Connectivity Applications

1. Start Python
2. Import the packages required
3. Open a connection to database
4. Create a cursor instance
5. Execute a query
6. Extract data
7. Clean up the environment

Step 2-import package

import mysql.connector

(or)

import mysql.connector as <alias_name>

Step 3 - open connection

connect() of mysql.connector package -

To establish connection to a MySQL database

Syntax:

<Connection-Object> = mysql.connector.connect

(host = <host-name>,user=<user name>,passwd=<password>[,database=<database>])

where :

user - username of MySQL

password - is the password of the user

host-name - the database server hostname or IP address

database - optional which provides the database name of MySQL database.

Example - Step 3

```
import mysql.connector as sq
mycon =
sq.connect(host='localhost',user='root',passwd='',database='Sample'
)
if mycon.is_connected():
    print('Successful')
```

Step 4 - create a cursor instance

A database cursor is a useful control structure of database connectivity.

From within a script/program, after establishing a connect to a database, the query gets sent to the server, where it is executed and the resultset is sent to the connection.

To access the resultset one row at a time a special type of control structure called *database cursor* can be created to access the records *row by row - database cursor is used.*

Step 4 - create a cursor instance

Syntax :

<cursor object> = <connectionobject>.cursor()

Example : cursor = mycon.cursor()

A **database cursor** is a special control structure that facilitates the row by row processing of records in resultset.

The **resultset** refers to a logical set of records that are fetched from the database by executing an SQL query and made available to the application program.

Step 5 - Execute SQL query

execute() - is used to execute SQL query

Syntax :

<cursorobject>.execute(<sql query>)

Example :

cursor.execute('select * from employee')

Step 6 - Extract data from Resultset

fetch() functions -

<data> = <cursor>.fetchall()	All records retrieved as per the query will be returned as a tuple form
<data> = <cursor>.fetchone()	Returns one record at a time (First record then the second and so on) when no records are available then None is returned.
<data> = <cursor>.fetchmany(<n>)	n records are returned as a tuple, when no records of n size of available it returns an empty tuple.
<variable> = <cursor>.rowcount	This is a property of cursor object it returns the number of rows retrieved from the cursor so far.

Step 7 - Clean up the environment

This is the final step to close the connection established.

Syntax :

<connection object>.close()

Example :

mycon.close()

SQL - TCL - Transaction Control Language commands

Commit - to permanently store the transactions of any addition/deletion/updation in the table.

Eg : cursor.commit()

rollback - to reverts the changes made by the current transaction (Note : after commit command rollback has no effect)

autocommit - like autosave feature of any application this can be set to True/False to perform auto commit of transactions into the table.

Example - fetchall()

```
import mysql.connector as sqltor
mycon =
    sqltor.connect(host='localhost',user='root',passwd='',database=Sample')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
cursor.execute('select * from sample')
data = cursor.fetchall()
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
```

Example-fetchmany(n)

```
import mysql.connector as sqltor
mycon =
    sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
cursor.execute('select * from sample')
data = cursor.fetchmany(2)
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
```

Example-fetchone()

```
import mysql.connector as sqltor
mycon
=sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
    print('Successful')
cursor=mycon.cursor()
cursor.execute('select * from sample')
data = cursor.fetchone()
count=cursor.rowcount
print('Total number of rows ',count)
print(data)
```

Example-fetchall() with criteria in query

```
import mysql.connector as sqltor
mycon =
    sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
cursor.execute('select * from sample where marks>90')
data = cursor.fetchall()
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
mycon.close()
```

Parameterised queries

1) x = 45

   select * from student where marks > x;

2)

x = ‘select * from student where marks={}’.format(45)

cursor.execute(x)

3)

x = ‘insert into student(stdno,name,marks)

values({},’{}’,{})’.format(1001,’Shaid’,89)

cursor.execute(x)

Example-formatted query

```
import mysql.connector as sqltor
mycon =
    sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
cursor.execute('select * from sample where marks>%s'%(90,))
data = cursor.fetchall()
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
mycon.close()
```

Example - updation in sql

```
import mysql.connector as sqltor
mycon = sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
name='SRIDHAR’'
rno = 3
inp=(name,rno)
query = “update sample set name =’%s’ where rno =%s”
cursor.execute(query,inp)
cursor.execute('select * from sample')
data = cursor.fetchall()
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
mycon.close()
```

Example - insertion in sql

```
import mysql.connector as sqltor
mycon = sqltor.connect(host='localhost',user='root',passwd='',database='rjk')
if mycon.is_connected():
   print('Successful')
cursor=mycon.cursor()
name='Sreejith’'
rno = 3
inp=(name,rno)
query = 'update sample set name = %s where rno =%s'
cursor.execute(query,inp)
st = "insert into sample(rno,name,marks,gender)
     values(%s,’%s’,%s,’%s’)"%(4,’Kumar’,67,’M’)
(or)
st = "insert into sample(rno,name,marks,gender)
     values({},'{}',{},'{}')".format(4,'Kumar',67,'M')
cursor.execute(st)
mycon.commit()
cursor.execute('select * from sample')
data = cursor.fetchall()
count=cursor.rowcount
print('Total number of rows ',count)
for row in data:
   print(row)
mycon.close()
```

General screen when through pip mysql-connector is installed

```
In [2]: pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading https://files.pythonhosted.org/packages/15/68/
c49e9a50dcf52f7ed03404d3f6ea81bff1f279ed9983c0f12f95beabb680/
mysql_connector_python-8.0.20-py2.py3-none-any.whl (358kB)
Collecting protobuf>=3.0.0 (from mysql-connector-python)
  Downloading https://files.pythonhosted.org/packages/a1/dd/
e391b3c63f728c07d0882f2825b95b74ee9337ecea0efb67c66c29c359b8/protobuf-3.12.0-
cp37-cp37m-win32.whl (907kB)
Requirement already satisfied: six>=1.9 in c:\users\welcome\anaconda3\lib
\site-packages (from protobuf>=3.0.0->mysql-connector-python) (1.12.0)
Requirement already satisfied: setuptools in c:\users\welcome\anaconda3\lib
\site-packages (from protobuf>=3.0.0->mysql-connector-python) (40.8.0)
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.20 protobuf-3.12.0
Note: you may need to restart the kernel to use updated packages.

In [3]: runfile('C:/Users/welcome/Desktop/rjk/rjk1805.py', wdir='C:/Users/
welcome/Desktop/rjk')
Successful
```