**Guided lab 342.3.2 - Insert Data into MySQL from Python**

# **Learning Objective:**

In this lab, we will discover how to insert data into a table using MySQL Connector/Python. By the end of this lab, learners will be able to insert data into MySQL using Python.

# Prerequisite:

You must be familiar with using MySQL languages like Data Manipulation Language (DML) and Data Query Language (DQL).

# Steps:

* Creating a database.
* Create a table for demonstration.
* To insert new rows into a MySQL table:

1. Connect to the MySQL database server by creating a new MySQLConnection object.
2. Initiate a MySQLCursor object from the MySQLConnection object.
3. Execute the INSERT statement to insert data into the table.
4. Close the database connection.

# Creating a Database for Demonstration.

For this lab, you must have the “**usersdb”** database. If you do not have the “**usersdb”** database setup, Run the below query on MySQL workbench.

| create database usersdb |
| --- |

# 

# Creating a Table for Demonstration.

Create a file named **createtable.py**, and add the following code:

| import mysql.connector as mydbconnection  from mysql.connector import Error  try:  conn = mydbconnection.connect(database='usersdb', user='root',password='password', port ='3306')  cursor=conn.cursor()  myquery2 = "CREATE TABLE `laptop` (`Id` int(11) NOT NULL,\  `Name` varchar(250) NOT NULL,\  `Price` float NOT NULL,\  `Purchase\_date` date NOT NULL)"  cursor.execute(myquery2)  print("Table is created")  except Error as e:  print("Failed tocreate table {}".format(e))  finally:  if conn.is\_connected():  conn.close()  print("MySQL connection is closed") |
| --- |

**Run your code.**

If you are not using a Jupyter notebook, open the command line interface, use the following command.

| python createtable.py |
| --- |

**Result:**

Table is created

MySQL connection is closed

# 

# Example 1: Insert a Single Row into MySQL Table from Python

In the below example we will Insert a Single Row into MySQL Table from Python.

Create a file named **insertsinglerecord.py**, and add the following code:

| import mysql.connector as mydbconnection  from mysql.connector import Error  try:  conn = mydbconnection.connect(database='usersdb', user='root',password='password', port='3305')  mySql\_insert\_query = """INSERT INTO Laptop (Id, Name, Price, Purchase\_date)  VALUES  (15, 'Lenovo ThinkPad P71', 6459, '2019-08-14') """  cursor = conn.cursor()  cursor.execute(mySql\_insert\_query)  conn.commit()  print(cursor.rowcount, "Record inserted successfully into Laptop table")  cursor.close()  except Error as e:  print("Failed to insert record into Laptop table {}".format(error))  finally:  if conn.is\_connected():  conn.close()  print("MySQL connection is closed") |
| --- |

**Run your code.**

If you are using a Jupyter notebook, open the command line interface and use the following command:

| python insertsinglerecord.py |
| --- |

Result:

1 Record inserted successfully into Laptop table

MySQL connection is closed

# Example 2: Use Python Variables in a MySQL Insert Query

Sometimes, you need to insert a Python variable value into a table’s column. For example, in the user signup form, the user enters his/her details. You can take those values in Python variables and insert them into a table.

* We can insert Python variables into the table using the prepared statement and parameterized query.
* Using a [parameterized query](https://pynative.com/python-mysql-execute-parameterized-query-using-prepared-statement/), we can pass Python variables as a query parameter in which placeholders (%s) are used for parameters.

Create a file named **insert\_record\_variable.py** and add the following code:

| import mysql.connector as mydbconnection  from mysql.connector import Error  def insert\_varibles\_into\_table(id, name, price, purchase\_date):  try:  conn = mydbconnection.connect(database='usersdb', user='root',password='password', port ='3306')  cursor = conn.cursor()  mySql\_insert\_query = """INSERT INTO Laptop (Id, Name, Price, Purchase\_date)VALUES (%s, %s, %s, %s) """  record = (id, name, price, purchase\_date)  cursor.execute(mySql\_insert\_query, record)  conn.commit()  print("Record inserted successfully into Laptop table")  except Error as error:  print("Failed to insert into MySQL table {}".format(error))  finally:  if conn.is\_connected():  cursor.close()  conn.close()  print("MySQL connection is closed")  insert\_varibles\_into\_table(2, 'Area 51M', 6999, '2019-04-14')  insert\_varibles\_into\_table(3, 'MacBook Pro', 2499, '2019-06-20') |
| --- |

**Run your code.**

If you are not using a Jupyter notebook, open the command line interface and use the following command:

| python insertsinglerecord.py |
| --- |

**Result:**

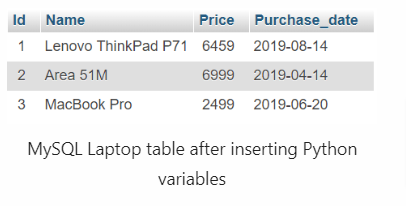
Record inserted successfully into Laptop table

MySQL connection is closed

Record inserted successfully into Laptop table

MySQL connection is closed

[Finished in 0.3s]



# Example 3: Insert multiple rows into MySQL Table using cursor’s executemany()

# In the previous example, we have used the execute() method of the cursor object to insert a single record.

# What if you want to insert multiple rows into a table in a single insert query from the Python application. Use the cursor’s executemany() function to insert multiple records into a table.

# Syntax of the executemany() method:

# cursor.executemany(operation, seq\_of\_params)

# This method executes Insert operation against all parameter sequences in the sequence seq\_of\_params argument.

# You need to include lists of tuples in the seq\_of\_params argument along with the insert query.

# Each tuple inside the list contains a single row that you want to insert so that you can add as many rows in the list and pass a list to a cursor.executemany() function along with the insert query.

# Note: Each tuple is enclosed within parentheses and is separated by commas. For example, to insert multiple rows in a laptop table, we can use the following SQL Query:

# INSERT INTO Laptop (Id, Name, Price, Purchase\_date) VALUES (%s, %s, %s, %s)

# 

# And in seq\_of\_params, we are passing the below List.

# records\_to\_insert = [(4, 'HP Pavilion Power', 1999, '2019-01-11'),

# (5, 'MSI WS75 9TL-496', 5799, '2019-02-27'),

# (6, 'Microsoft Surface', 2330, '2019-07-23')]

Create a file named **insert\_multipleRecords.py**. Add the following code:

| import mysql.connector as mydbconnection  from mysql.connector import Error  try:  conn = mydbconnection.connect(database='usersdb', user='root',password='password', port ='3306')  mySql\_insert\_query = """INSERT INTO Laptop (Id, Name, Price, Purchase\_date)  VALUES (%s, %s, %s, %s) """  records\_to\_insert = [(4, 'HP Pavilion Power', 1999, '2019-01-11'),  (5, 'MSI WS75 9TL-496', 5799, '2019-02-27'),  (6, 'Microsoft Surface', 2330, '2019-07-23')]  cursor = conn.cursor()  cursor.executemany(mySql\_insert\_query, records\_to\_insert)  conn.commit()  print(cursor.rowcount, "Record inserted successfully into Laptop table")  except Error as error:  print("Failed to insert record into MySQL table {}".format(error))  finally:  if conn.is\_connected():  cursor.close()  conn.close()  print("MySQL connection is closed") |
| --- |

**Run your code.**

If you are not using a Jupyter notebook, open the command line interface and use the following command:

| python insert\_multipleRecords.py |
| --- |

**Result:**

3 Record inserted successfully into Laptop table

MySQL connection is closed

