



Unit 13

Database Connectivity

Table of Contents

13.1	Introduction	
	Learning Objectives	
13.2	Python SQL	
13.2.1	Environmental Setup	
13.2.2	Install mysql.connector	
	Self-Assessment Questions	
13.3	Creating the Connection	
13.3.1	Creating Cursor Object	
	Self-Assessment Questions	
13.4	Fetching the Data	
13.4.1	Creating the Table	
13.4.2	Alter Table	
	Self-Assessment Questions	
13.5	Insert Operations	
13.5.1	Read Operation	
13.5.2	Update Operation	
13.5.3	Join Operation	
	Self-Assessment Questions	
13.6	Summary	
13.7	Glossary	
13.8	Case study	
13.9	Terminal Questions	
	Answer Keys	
	A. Self Assessments Questions	
	B. Terminal Questions	
13.10	Suggested Books and e-References	

13.1 INTRODUCTION

In the past units of the book, you learned how to work with Python and its basic concepts. You learned how to solve different problems using Python and how to use the various features and functions offered by Python. However, to ensure that the immense capability of Python is completely utilized, you need to use it for website development. Programmers find Python a handy and powerful tool when they require a strong base for web development.

To make use of Python in real life, you will need to learn to manage data in Python. The code that you program using the language should be able to accept a large amount of data, process it and show the result in a comprehensible and dynamic form. When it comes to the web, a programmer needs to manage data to offer any service. Take an example of an emailing website. When users access the website, they will have to provide their User ID or Email ID along with a unique password. The code should be able to verify if the credentials match the ones that were recorded previously.

This previously recorded data should be arranged in the form of a database. The database ensures that the entire information essential for the working of the website is well-arranged and formatted. This increases the accessibility of the data stored in it as well as the functionality of the website.

What is data? Data is arranged information. It eases the computation and efficiency of the computer. A database is an organized collection of data, usually arranged in rows and columns that can be easily accessed, assessed, and managed. Its primary function is to store large data, retrieve it as per instructions, and manage it. There are various software available which helps you create and maintain data such as MySQL, Sybase, Oracle, Informix and more.

Here are some of the terms that you should be aware of before diving deeper into the unit. These are common terms used when referencing data and database.

Database Management System: A database management system or DBMS is software that is used to store, retrieve, define, and manage the data available in a database. These are modern software that could analyses the data and provide results accordingly.

SQL: Structured Query Language comprises commands that are used to manipulate and operate the data stored in the database. It depends upon relational algebra and tuple relational calculus.

RDBMS: Relational Database Management System was introduced by E.F. Codd. It includes elements such as a table, tuples, instance, schema, keys, and attributes. It allows you to define a relationship between the tables present in the database.

Python Database API: It is the standard way through which you can define how different Python modules can be used to access databases.

Through Python Database API, you can easily understand the working of the modules, how to port code across different databases and simplify the process of database connectivity in Python. Here is a list of sections that comprise the Python Database API interface.

- Module Interface
- Connection Objects
- Cursor Objects
- Type Objects and Constructors
- Implementation Hints

STUDY NOTE

There are various database servers supported by Python. MySQL is used more commonly for its varied advantages.

Table: When working with a database, a table looks like a spreadsheet and acts as a matrix with data.

Column: A column contains the same kind of data. It is also called a data element.

Row: It is a group of related data. One can refer to it as a tuple, entry, or data.

Redundancy: When the same data is stored more than once.

Primary key: A key value is something that does not occur twice in the table. It is unique.

Foreign key: It is the data that links two tables.

Compound Key: A compound key consists of multiple columns as defining one column is not unique.

Index: An index is the list of all data and topics present in the database.

Referential integrity: It ensures that a foreign key always points to an existing row.

LEARNING OBJECTIVES

After studying the chapter, you will be able to:

- ❖ Understand how to install Python SQL and setting up mysql.connector.
- ❖ Describe how to connect to a Python SQL database and create a cursor object.
- ❖ Know how to create and alter a table as per the requirements.
- ❖ Understand how to use insert operation, update operation, and join operation.



13.2 SYSTEM REQUIREMENTS

What is MySQL?

It is an RDBMS with a multitude of features. You can create multiple related tables in a database using MySQL. Being open-source software, it can be downloaded freely and customised as per your needs. It has built-in support for languages such as Python and PHP. It offers an efficient security system and consists of a thread-based memory allocation system. There is no risk of memory leakage with MySQL. It can also hold large databases with efficiency.

STUDY NOTE

MySQL is developed and maintained by a company called MySQL DB.

You can integrate MySQL database in Python using the module names MySQL Connector Python. Using the MySQL database server, you can develop and integrate Python applications. There are various other modules that one can choose as per their requirements such as PyMySQL, MySQL DB, OurSQL, and more. Among the various available options, we will choose to continue with MySQL Connector. Note that the syntax, method, and ways of accessing the database are some of the factors that are the same for all the modules. This is because they are designed to adhere to the regulations of Python Database AP V2.0.

Advantages of MySQL Connector Python

There are various benefits of choosing the module MySQL Connector Python over others for its better support and efficiency. These benefits are listed below.

- MySQL Connector Python is coded in Python and can execute queries through Python seamlessly.
- Supported by Oracle, it is the official link between MySQL and Python.
- MySQL Connector Python and frequently updated and maintained for the best experience and result.

13.2.1 ENVIRONMENTAL SETUP

The system requirements for Python MySQL are similar to that of Python. By now, you must have set up Python on your system. To access the MySQL database, the first requirement for Python is to download a MySQL driver. Here, we are using MySQL Connector as the driver. To download it, you will need PIP.

PIP is pre-installed with the Python package and will be readily available for use. You will need root or administrative access to begin the installation process. MySQL Connector needs to be in the system's path. Unless it can find Python on your system, it will not install.

In Unix, Python is located in a directory included in the default path. However in Windows, Python may not exist in the system's path. It can be added to the directory containing python.exe by the user.

The steps that are provided below to install Python MySQL Connector are applicable for the following specifications.

Platforms: 64-bit Windows, Windows 10, Windows 7, Windows 8, Windows Vista, Windows XP, Linux, Ubuntu Linux, Debian Linux, SUSE Linux, Red Hat Linux, Fedora, MacOS.

Python Version: Python 2 and Python 3.

MySQL Version: Version 4.1 or higher.

13.2.2 INSTALL MYSQL.CONNECTOR

MySQL Connector can be installed in various ways and methods. These are explained in detail below.

Using Pip Command to Install MySQL Connector Python

It is a straightforward process to download MySQL Connector Python with pip in Python. The MySQL Connector Python is available in pypi.org and can be downloaded using the pip command. On the command window, type the command given below.

```
pip install mysql – connector – python
```

You can also mention the version of the module in the command line if you are facing any issues while downloading the module. The command with version will be as provided below.

```
pip install mysql – connector – python == 8.0.11
```

Once the command is run, you will need to verify that MySQL Connector Python has been successfully installed. You can do this by checking the following messages that appear on the command window.

```
Connecting mysql – connector – python.
```

```
Downloading packages.
```

```
Requirement already satisfied: setup tools in D:  
python\python 37-32\lib\site – packages
```

```
Installing collected packages: mysql – connector – python.
```

```
Successfully installed mysql – connector – python – 8.0.13.
```

STUDY NOTE

Oracle acquired Sun Microsystems and MySQL in 2010. MariaDB was created with MySQL code after its acquisition to keep it free.

Using Source Code Distribution (On Windows)

Firstly, go to the link <https://dev.mysql.com/downloads/connector/python/> to download the MySQL Connector Python for Windows for free. It will be available in the .zip extension. The link leads you to the latest version of the module.

You can also download older versions that are compatible with your device from the website at which you land when you click on the link provided below.

Click on the download button to save the ZIP file on your system.

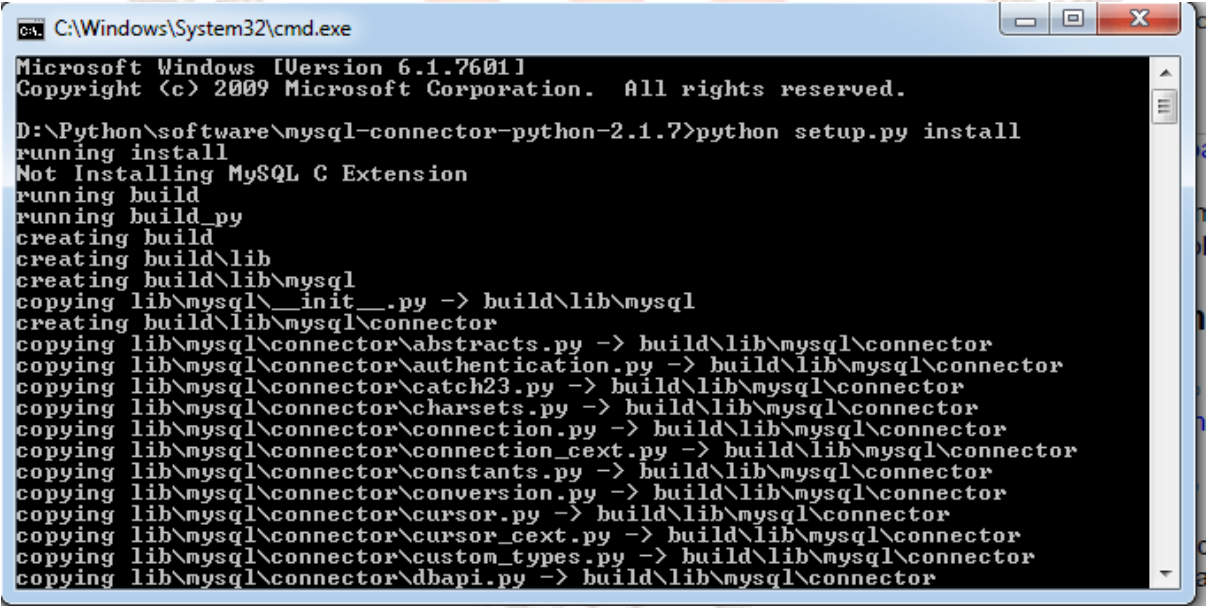
Unzip the zip archive in the directory of your choice. You can use any tool for this process.

Start a console window and change the location to the folder or directory where you unzipped the Zip archive.

Use the command provided below to begin installation inside the MySQL Connector Python folder.

C:\> python setup.py install.

Here, we have considered that the zip file is saved in the C drive of the system. Now, your screen should look like the one shown in the image below.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

D:\Python\software\mysql-connector-python-2.1.7>python setup.py install
running install
Not Installing MySQL C Extension
running build
running build_py
creating build
creating build\lib
creating build\lib\mysql
copying lib\mysql\__init__.py -> build\lib\mysql
creating build\lib\mysql\connector
copying lib\mysql\connector\abstracts.py -> build\lib\mysql\connector
copying lib\mysql\connector\authentication.py -> build\lib\mysql\connector
copying lib\mysql\connector\catch23.py -> build\lib\mysql\connector
copying lib\mysql\connector\charsets.py -> build\lib\mysql\connector
copying lib\mysql\connector\connection.py -> build\lib\mysql\connector
copying lib\mysql\connector\connection_cext.py -> build\lib\mysql\connector
copying lib\mysql\connector\constants.py -> build\lib\mysql\connector
copying lib\mysql\connector\conversion.py -> build\lib\mysql\connector
copying lib\mysql\connector\cursor.py -> build\lib\mysql\connector
copying lib\mysql\connector\cursor_cext.py -> build\lib\mysql\connector
copying lib\mysql\connector\custom_types.py -> build\lib\mysql\connector
copying lib\mysql\connector\dbapi.py -> build\lib\mysql\connector
```

Source: Pynative

Fig 1: Python Setup in Windows

Download MySQL Connector Python on Linux

Using Source Code Distribution

Go to the link <https://dev.mysql.com/downloads/connector/python/> and download the suitable version of MySQL Connector Python on your system.

You can choose the version of the TAR archive as per the specifications of your system.

Once the download is complete, you will need to follow the steps provided below to install the module.

To untar the downloaded file, use the command provided below.

```
shell>tar xzf mysql – connector – python – VERSION.tar.gz
```

Change the directory where you extracted the tar file.

```
shell> cd mysql – connector – python – VERSION
```

Now, execute the command provided below to install the module on Linux.

```
shell> sudo python setup.py install
```

Download MySQL Connector Python on macOS

Firstly, download the source code for macOS from the link <https://dev.mysql.com/downloads/connector/python/>.

Now, download the mysql-connector-python-8.0.11-macos10.13.dmg file.

Install the module by opening the downloaded file and double-clicking on the .pkg file.

Download MySQL Connector Python on Ubuntu

Use the code provided below to install MySQL Connector Python.

```
sudo apt-get install mysql – connector – python
```

```
pip install mysql – connector – python
```

If the above code does not work, then you can proceed by downloading the Source Code from <https://dev.mysql.com/downloads/connector/python/> link.

You will get two DEB packages. Download them both and unpack them. To install them, use the code provided below.

```
sudo dpkg -I /path_to_downloaded_deb_file
```

```
sudo apt-get install -f
```


When you run both these commands one after the other, the MySQL Connector Python will install on Ubuntu.

To test if the installation was successful, you can run the command provided below.

Import mysql.connector

If the code runs without any errors, then you have the module installed on your system.

ACTIVITY 1

Try installing the MySQL Connector Python Module in different environments. Compare their user interface and other surficial features. Note how changing the environment can affect the working of the module.

SELF-ASSESSMENT QUESTIONS

1. For the Ubuntu environment, you will need to download the source code having the extension _____.
2. _____ command can be used to verify that the MySQL Connector Python module is successfully installed in the system.
3. Which of the following is the easiest way to download the module for MySQL Connector Python?
 - A. Downloading source code
 - B. Pip method
 - C. MySQL Database
 - D. None of the above
4. 5. The MySQL Connector Python module is usually already available with the Python package. (True/False)

You cannot download MySQL Connector Python without Python already installed in the environment. (True/False)

13.3 CREATING THE CONNECTION

The first step after downloading the module is to establish the connection between the database. You can create the database using the create command in the command prompt. This command comes with the MySQL module and is built-in. Here is an example.

```
#mysqladmin create Employee
```

This command will create a database named employee.

You can also create a database using Python with the help of the code provided below.

```
import mysql.connector

db1 = mysql.connector.connect (
    host = "localhost",
    user = "myusername",
    password = "mypassword"
)
mycursor = db1.cursor ()
mycursor.execute ("CREATE DATABASE mydatabase")
```

In this code, you will find a cursor method. This will be explained in detail later in the unit.

To create a connection with an existing database, here is the code that you will need to type on your Python window.

```
import mysql-connector
mydatabase = mysql.connector.connect (
    host = "localhost",
    user = "myname",
    password = "mypassword"
)
print (mydatabase)
```

Here, the name of the database is mydatabase. The username is myusername and the password is mypassword. We are using the constructor connect () that accepts username, password, host, and the name of the password. It will return an object of the MySQL Connection class.

You can disconnect the database using the exit command in the mysql prompt. This is applied as follows.

```
mysql> exit
```

Or, you can close the connection through the Python window by adding the code provided below.

```
mydatabase.close ()
```

13.3.1 CREATING CURSOR OBJECT

You can create a cursor object using the `cursor ()` method that is built-in in the MySQL Connector Python module. The cursor objects are used to process and analyse individual rows returned by database system queries. Through the cursor, you can manipulate the whole result set at once. The code to create a cursor object using Python is provided below.

```
import mysql.connector
conn = mysql.connector.connect (
    host = "localhost",
    user = "myusername"
    password = "mypassword"
    database = "db1"
)
cursor = conn.cursor ()
```

The cursor class has the following properties.

- **column_names:** It returns the list containing the column names of a result. It is a read-only value.
- **description:** It returns a list of description of columns in a result rest. It is read-only as well.
- **lastrowid:** It returns the value generated in an auto-incremented column of the table in the last INSERT or UPDATE operation.
- **rowcount:** It returns the number of rows that undergone the SELECT or UPDATE operations.
- **statement:** It returns the last executed statement.

STUDY NOTE

Other than the `fetchall ()` method, you can also use the `fetchone ()` method to retrieve data. It retrieves the next row of a query result set.

SELF-ASSESSMENT QUESTIONS

6. _____ is the method used to analyse individual rows in a database system.
7. A connect () method accepts _____, _____, and _____.
8. Through which of the following property can you get the last executed statement in a database?
 - a. last_statement
 - b. statement
 - c. statement_db1
 - d. laststatement
9. column-name returns a value that is read-only. (True or False)
10. Once connected, you cannot disconnect to a MySQL database. (True or False)

13.4 FETCHING THE DATA

In this section, we will learn the basics of manipulating with database in MySQL using Python. If a database already exists, using few code lines, you can fetch the data present in the database. This can be done by either retrieving rows or columns of the table. You can also specify the data that you want to retrieve using queries such as WHERE.

Go through the code provided below. This code fetches the data present in a table using the query SELECT. Note that all the queries are written in all caps. The code uses the method fetchall () to retrieve all the rows from a MySQL table.

```
import mysql.connector
conn = mysql.connector.connect (
    host = "localhost",
    user = "myusername",
    password = "mypassword",
    database = "db1"
)
my_database = conn.cursor ()
sql_table = "SELECT * FROM EMPLOYEE"
my_database.execute (sql_table)
output = my_database.fetchall ()
```

```
for x in output:  
    print (x)
```

Here, the `execute ()` method is used to execute the MySQL query. The variables retrieved by the `SELECT` query have been saved in the variable `sql_table`. The result set is saved as a list of tuples with the help of `fetchall ()`.

13.4.1 CREATING A TABLE

To create a table in MySQL, you use the `CREATE` query. Through Python, we will use this query and create a new table in our database. The code to form a new table in the database is provided below. Ensure that the name of the database has been defined while forming the connection with it.

```
import mysql.connector  
db1 = mysql.connector.connect (  
    host = "localhost",  
    user = "myusername",  
    password = "mypassword",  
    database = "my_database"  
)  
mycursor = db1.cursor()  
mycursor.execute (" CREATE TABLE employee (name VARCHAR (200), address  
VARCHAR (200))")
```

The code creates a table called `employee` in the database named `db1`. The table contains two columns, one will store the names of the employees and the other will store the address. Note that the data type for these two columns is set to be `VARCHAR`, which is a character string with a length of 200 bits.

You should also create a primary key while creating a table. The primary key is a column with a unique key for each record. The example to create a primary key has been provided with the code below.

```
import mysql.connector  
db1 = mysql.connector.connect (  
    host = "localhost",  
    user = "myusername",  
    password = "mypassword",  
    database = "my_database"  
)  
mycursor = db1.cursor()  
mycursor.execute (" CREATE TABLE employee (id INT AUTO_INCREMENT PRIMARY  
KEY, name VARCHAR (200), address VARCHAR (200))")
```

Here is a list of data types available in MySQL that can help you create various tables that can store different type of data.

TABLE 1: DATA TYPES AVAILABLE IN MYSQL

Data Type	Description
NUMERIC	It represents a number.
DECIMAL (precision, scale)	It has two parts, precision and scale. Precision specifies the number of decimal digits and scale specifies the number of digits that are to be stored following the decimal point.
INT	It represents an integer ranging from -2147483648 to 2147483647.
SMALLINT	They also represent integers from -32768 to 32767.
FLOAT	It represents a number from -1.79E+308 to 1.79E+38.
REAL	It represents a number ranging from -3.40E+38 to 3.40E+38.
DOUBLE PRECISION	It represents a 64-bit value from 10^{308} to 10^{-323}
DATETIME	It represents date in the format of yyyy-mm-dd and time in hh:mm:ss.
DATE	It shows date in the format of yyyy-mm-dd.
TIMESTAMP	It represents dates ranging from 1970 to 2037 with a resolution of one second.
TIME	Shows time in the format of hh:mm:ss.
YEAR	It represents the year in form of yyyy from 1901 to 2155.
CHAR (no-of-bytes)	It represents strings of the length from 0 to 255.
VARCHAR (no-of-bytes)	It can be used to store only that many characters as needed. Thus, it is a variable string of characters.
TEXT	It stores strings without the need of specification of size.

13.4.2 ALTER TABLE

ALTER query is used to change an existing table. When working in MySQL, you can simply apply the query and add a column to the table. An example is shown below.
ALTER TABLE employee MODIFY COLUMN address VARCHAR (255).

Thus, the length of the string that could be stored in the address column is changed from 200 to 255. In Python, this code will be as shown below.

```
import mysql.connector
db1 = mysql.connector.connect (
    host = "localhost",
    user = "myusername",
    password = "mypassword",
    database = "my_database"
)
mycursor = db1.cursor()
mycursor.execute (" ALTER TABLE employee ADD COLUMN id INT AUTO_INCREMENT
PRIMARY KEY")
```

The program above will add a column for the primary key in the table employee.

STUDY NOTE

rollback () method is used to roll back a database to a pending transaction and start again.

SELF-ASSESSMENT QUESTIONS

11. _____ are commands used in MySQL to perform an operation.
12. _____ is a column with values that uniquely identify each data stored in the table.
13. Which of the following query will you use to change a column in a pre-existing table?
 - A. MODIFY
 - B. ALTER
 - C. CHANGE
 - D. None of the above.
14. Which of the following data type in MySQL can be used to store strings of an undefined length?
 - A. TEXT
 - B. STRING
 - C. CHAR
 - D. VARCHAR
15. You can store decimal values in a MySQL table using DECIMAL data type. (True/False)

13.5 INSERT OPERATION

With INSERT Operation, you can add new rows to an existing table. You will need to specify the name of the table, column names, and values that you need to store in the added row.

The execute () method is used to accept a query as a parameter. The method executes the given query in Python. You must have noticed the function in various examples given previously. The code provided below gives an example regarding how the INSERT query works.

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
```

STUDY NOTE

To commit any pending transactions in a database, the `commit()` method is used. Thus, whenever you update a table, the `commit()` function is immediately executed to save the updates on the disk.

```
user="yourusername",
password="yourpassword",
database="mydatabase"
)
mycursor = mydb.cursor()
sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)
mydb.commit()
print (mycursor.rowcount, "record inserted.")
```

You can insert multiple rows into a table using the method `executemany()`. You will need to provide parameters such as a list of tuples with the method. The list should contain the data that you want to insert.

13.5.1 READ OPERATION

To read values saved in the database, the `SELECT` query is used. There are various methods in Python that return the data stored in the table. These methods include `fetchall()` and `fetchone()`. These methods are already discussed above. You can use the `SELECT` query to select a complete table, some columns from the table, or only a row. Here is an example using the `fetchone()` method to return only one row.

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM customers")
myresult = mycursor.fetchone()
print(myresult)
```

To select data from a table, one can apply the filter using the `WHERE` filter. Here is what the command will be like when using the query.

```
sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"
```

13.5.2 UPDATE OPERATION

Through UPDATE operation, the databases can be updated. You can update specific rows using the WHERE clause.

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)
mycursor = mydb.cursor()
sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) affected")
```

13.5.3 JOIN OPERATION

To retrieve data that has been divided into two tables, the JOIN operation is used. The working of the operation has been depicted through the program provided below.

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)
mycursor = mydb.cursor()
sql = "SELECT \
    users.name AS user, \
    products.name AS favorite \
    FROM users \
    INNER JOIN products ON users.fav = products.id"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for x in myresult:
    print(x)
```

INNER JOIN only retrieves values that match in both the tables. If you need to retrieve all the values, then you can use LEFT JOIN and RIGHT JOIN statement.

SELF-ASSESSMENT QUESTIONS

16. _____ clause is used to specify the row from where a data is to be manipulated.
17. _____ method is required to save the pending transactions made in a database.
18. Which of the following operation is used to retrieve data from more than one table at a time?
- A. JOIN
 - B. SELECT
 - C. FIND
 - D. ALL TABLES
19. To add a new row into an existing table, which operation is used?
- A. ADD
 - B. UPDATE
 - C. INSERT
 - D. JOIN
20. Data from two separate tables can be combined using MySQL Connector Python. (True/False)



13.6 SUMMARY

- Databases are a way one can manage, structure, and organise a large amount of data.
- A database management system is software used to manage a database that is in the form of tables with rows and columns.
- When a DBMS retrieves information by using the data in the table, then it is called a relational database management system (RDBMS).
- Python Database API is the process through which different modules can access databases.
- MySQL comes with the Python package. However, it can be installed separately as well.
- Various commands can be used in MySQL for server administration. Create command is used to create a new database.
- Various other commands can be used to access, modify, and manage the tables in the database. These include creating, select, insert, update, delete, and others.
- One can use various data types in MySQL. These are Numeric, Decimal, Int, Smallint, Float, Real, Double Precision, DateTime, Date, TimeStamp, Time, Year, Char, Varchar, and Text.
- One needs to connect to a database to Python before manipulating it using Python code.
- connect () method is used to connect to a database.
- cursor () method returns a new cursor object and represents a database cursor.
- Various methods can be used in Python to extract records from the result set. These are execute (), executemany (), fetchone (), and fetchall ().

13.7. GLOSSARY

Database: A repository that stores related information.

Database Management System: A software mechanism to access, retrieve and manage data present in a database.

MySQL: An RDBMS that can be integrated with Python to manage and create a database.

Query: Commands used in MySQL to manipulate and manage the data in the database.

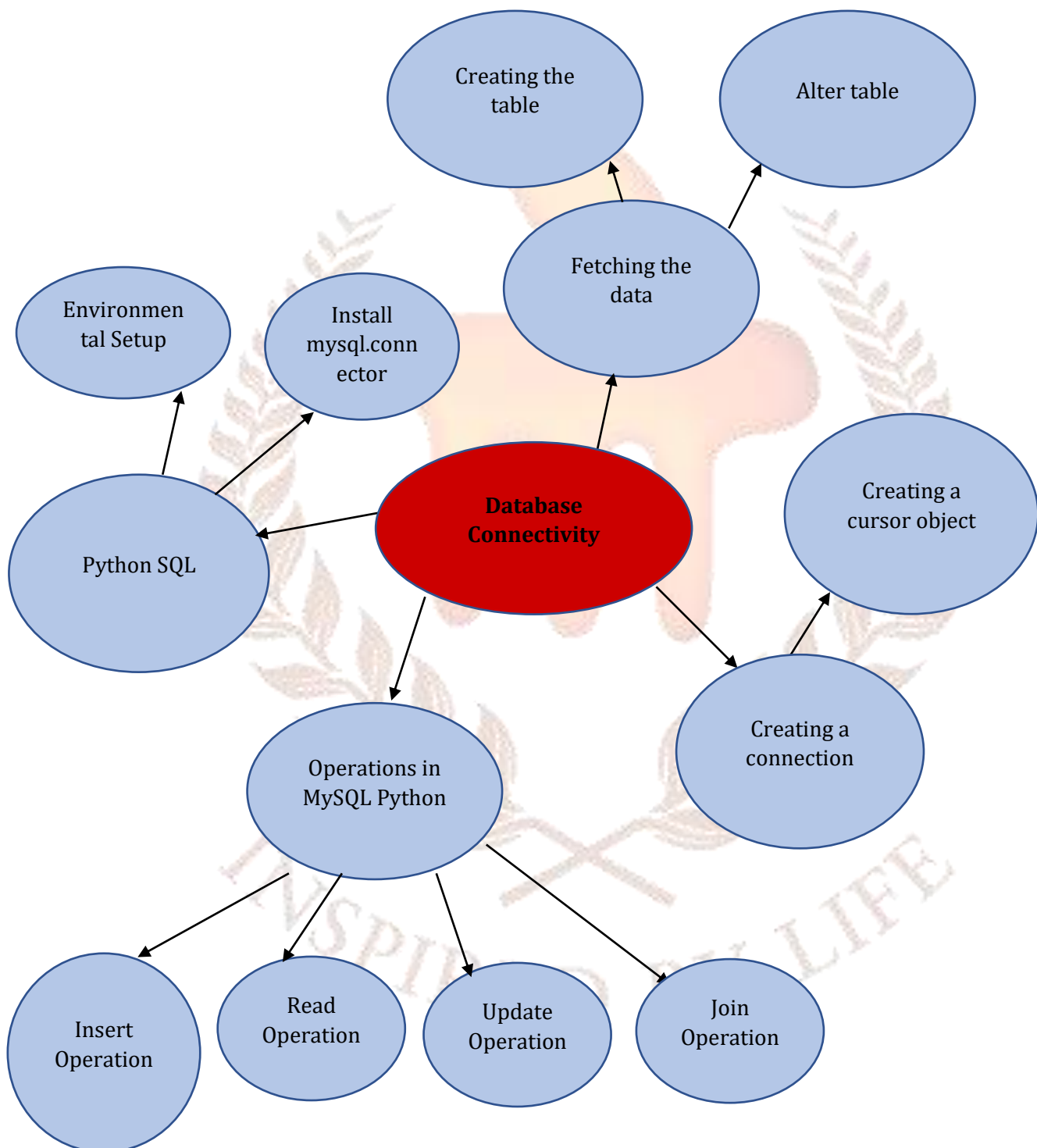


Fig 2: Conceptual Map

13.8 CASE STUDY**Online registration process in CBSE School using Python**

A school is set to begin an online registration process to keep track of all the students that are or will be enrolled in the school. The students will be able to access an online registration form that receives their basic information such as name, date of birth, name of guardians, permanent address, current address, enrolment number, blood group, etc.

New students can fill in all this information. The form is created such that students can update and alter the details that they previously entered. Such that, someone's current address has changed, so they can change the address in the form and update their information in the database.

Consider yourself a part of the technical team that has been set to code the registration form. You will need to create identify the elements of the table that are needed to be stored as registration details. Next, you will need to connect to the database and create a table. The code created should be able to store the information entered by the students in the respective columns of the tables. Also, alterations and modifications should be allowed in the table. The primary key for each student will be their enrolment number. It is unique for each student admitted into the school. Once the table has been coded and created, you need to verify the data stored in the database.

Based on the information provided and the tasks needed to execute the project, answer the questions provided below.

Source: IndiaNic.com

Questions:

1. What will be the columns that you will use as different keys needed in a database?
2. How will you proceed with the verification of data stored in the database?

13.9 TERMINAL QUESTIONS**SHORT ANSWER QUESTIONS**

- Q1. Write the steps to install `mysql.connector` using source code in a Windows environment.
- Q2. How can you modify the rows of an existing table using MySQL Connector Python?
- Q3. Write a Python program to delete a row from a table.
- Q4. What are the data types available in MySQL? Describe them.
- Q5. Write a Python program to retrieve required data from a table.

LONG ANSWER QUESTIONS

- Q1. Write a Python program to connect to the database driver and print its version.
- Q2. Write a Python program to create a database connection to a database that already resides in the memory.
- Q3. Write a Python program that fetches information regarding the name of the hospital and doctor using hospital ID and doctor ID in a table containing such information.
- Q4. Write a Python program that gives a list of doctors as per their speciality.
- Q5. Write a Python program that updates the number of years of experience of a doctor in the table.

ANSWERS**SELF ASSESSMENT QUESTIONS**

1. DEB
2. `import mysql.connector`
3. B. pip method
4. True
5. True
6. `cursor ()`
7. Host, username, and password.
8. Statement
9. True
10. False
11. Query
12. Primary Key
13. B. ALTER
14. A. TEXT
15. True
16. WHERE
17. `commit ()`
18. A. JOIN

- 19. C. INSERT
- 20. True

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Answer 1: Here are the steps to install MySQL Connector Python to Windows.
Download the ZIP file of the source code on your system.
Unzip the zip archive in the directory of your choice.
Start a console window and change the location to the folder or directory where you unzipped the Zip archive.
Use the command provided below to begin installation inside the MySQL Connector Python folder.
C:\> python setup.py install.

Answer 2: To modify the rows in the table of a database, one can use the ALTER query. One will need to define the name of the table, the name of the row, and the data that needs to be changed.

Answer 3:

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)
mycursor = mydb.cursor()
sql = "DELETE FROM customers WHERE address = 'Mountain 21'"
mycursor.execute(sql)
mydb.commit()
print (mycursor.rowcount, "record(s) deleted")
```

Answer 4: There are various data types one can use in MySQL for variables. These are explained in the table below.

Data Type	Description
NUMERIC	It represents a number.
DECIMAL (precision, scale)	It has two parts, precision and scale. Precision specifies the number of decimal digits and scale specifies the number of digits that are to be stored following the decimal point.
INT	It represents an integer ranging from -2147483648 to 2147483647.
SMALLINT	They also represent integers from -32768 to 32767.
FLOAT	It represents a number from -1.79E+308 to 1.79E+38.
REAL	It represents a number ranging from -3.40E+38 to 3.40E+38.

```
Answer 5: import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM customers")
myresult = mycursor.fetchall()
for x in myresult:
    print(x)
```

LONG ANSWER QUESTIONS

Answer 1:

```
import mysql.connector
def get_connection():
    connection = mysql.connector.connect(host='localhost',

    database='python_db',
        user='pynative',

    password='pynative@#29')
    return connection

def close_connection(connection):
    if connection:
        connection.close()

def read_database_version():
    try:
        connection = get_connection()
        cursor = connection.cursor()
        cursor.execute("SELECT version();")
        db_version = cursor.fetchone()
        print ("You are connected to MySQL version: ", db_version)
        close_connection(connection)
    except (Exception, mysql.connector.Error) as error:
        print ("Error while getting data", error)

print ("Question 1: Print Database version")
read_database_version()
```

Answer 2:

```
import mysql.connector
try:
    mysql_Connection = mysql.connector.connect('temp.db')
```

```
conn = mysql.connector.connect(': memory:')
print ("\nMemory database created and connected to MySQL.")
mysql_select_Query = "select mysql_version();"
conn.execute(mysql_select_Query)
print ("\nmysql Database Version is: ", mysql.connector.version)
conn.close()
except mysql.connector.Error as error:
    print ("\nError while connecting to mysql", error)
finally:
    if (mysql_Connection):
        mysql_Connection.close()
        print ("\nThe MySQL connection is closed.")
```

Answer 3:

```
import mysql.connector

def get_connection():
    connection = mysql.connector.connect(host='localhost',
                                         database='python_db',

user='pynative',
                                         password='pynative@#29')
    return connection

def close_connection(connection):
    if connection:
        connection.close()

def get_hospital_detail(hospital_id):
    try:
        connection = get_connection()
        cursor = connection.cursor()
        select_query = """select * from Hospital where Hospital_Id = %s"""
        cursor.execute(select_query, (hospital_id,))
        records = cursor.fetchall()
        print ("Printing Hospital record")
        for row in records:
            print ("Hospital Id:", row [0], )
            print ("Hospital Name:", row [1])
            print ("Bed Count:", row [2])
        close_connection(connection)
    except (Exception, mysql.connector.Error) as error:
        print ("Error while getting data", error)

def get_doctor_detail(doctor_id):
    try:
        connection = get_connection()
```



```
cursor = connection.cursor()
select_query = """select * from Doctor where Doctor_Id = %s"""
cursor.execute(select_query, (doctor_id,))
records = cursor.fetchall()
print ("Printing Doctor record")
for row in records:
    print ("Doctor Id:", row [0])
    print ("Doctor Name:", row [1])
    print ("Hospital Id:", row [2])
    print ("Joining Date:", row [3])
    print ("Specialty:", row [4])
    print ("Salary:", row [5])
    print ("Experience:", row [6])
close_connection(connection)
except (Exception, mysql.connector.Error) as error:
    print ("Error while getting data", error)
```

```
print ("Question 2: Read given hospital and doctor details \n")
get_hospital_detail(2)
print("\n")
get_doctor_detail(105)
```

Answer 4:

```
import mysql.connector
def get_connection():
    connection = mysql.connector.connect(host='localhost',
                                         database='python_db',
                                         user='pynative',
                                         password='pynative@#29')
    return connection

def close_connection(connection):
    if connection:
        connection.close()

def get_specialist_doctors_list(speciality, salary):
    try:
        connection = get_connection()
        cursor = connection.cursor()
        sql_select_query = """select * from Doctor where Speciality=%s and Salary > %s"""
        cursor.execute(sql_select_query, (speciality, salary))
        records = cursor.fetchall()
        print ("Printing doctors whose specialty is", speciality, "and salary greater than",
        salary, "\n")
        for row in records:
            print ("Doctor Id: ", row [0])
            print ("Doctor Name:", row [1])
```

```

    print ("Hospital Id:", row [2])
    print ("Joining Date:", row [3])
    print ("Specialty:", row [4])
    print ("Salary:", row [5])
    print ("Experience:", row [6], "\n")
    close_connection(connection)
except (Exception, mysql.connector.Error) as error:
    print ("Error while getting data", error)

```

```

print ("Question 3: Get Doctors as per given Speciality\n")
get_specialist_doctors_list("Garnacologist", 30000)

```

Answer 5:

```

import mysql.connector
import datetime
from dateutil.relativedelta import relativedelta

def get_connection():
    connection = mysql.connector.connect(host='localhost',
                                         database='python_db',

user='pynative',

password='pynative@#29')
    return connection

def close_connection(connection):
    if connection:
        connection.close()

def update_doctor_experience(doctor_id):
    try:
        connection = get_connection()
        cursor = connection.cursor()
        select_query = """select Joining_Date from Doctor where Doctor_Id = %s"""
        cursor.execute(select_query, (doctor_id,))
        joining_date = cursor.fetchone()
        joining_date_1 = datetime.datetime.strptime(''.join(map(str, joining_date)), '%Y-%m-%d')
        today_date = datetime.datetime.now()
        experience = relativedelta(today_date, joining_date_1).years
        connection = get_connection()
        cursor = connection.cursor()
        sql_select_query = """update Doctor set Experience = %s where Doctor_Id = %s"""
        cursor.execute(sql_select_query, (experience, doctor_id))
        connection.commit()
        print ("Doctor Id:", doctor_id, " Experience updated to ", experience, " years")
        close_connection(connection)

```

```
except (Exception, mysql.connector.Error) as error:  
    print ("Error while getting doctor's data", error)  
  
print ("Question 4: Calculate and Update experience of all doctors \n")  
update_doctor_experience(101)
```

13.10 SUGGESTED BOOKS AND E-REFERENCES

Books:

1. Eric Matthes (2016), Python Crash Course: A Hands-On, Project-Based Introduction to Programming. Packt Publishing ltd.,
2. John M. Zelle (2009), Python Programming: An Introduction to Computer Science (Preliminary Second Edition).
3. Mark Lutz (2011), Python Programming: A Powerful Object-Oriented Programming (Fourth Edition).
4. Sebastian Raschka (2017), Python Machine Learning - Machine Learning and Deep Learning with Python (Edition 2)

E-References

- Python Programming Certification Training Course, last viewed on March 25, 2021, < <https://www.edureka.co/python-programming-certification-training> >
- Python Tutorials and Sample Programs, last viewed on March 25, 2021, < <https://www.w3schools.com/python/> >
- Integrated Development Environments, last viewed on March 25, 2021, < <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments> >