What is Data?

Data is a collection of a distinct unit of information. This “data” is used in a variety of forms of text, numbers, media and many more. Talking in terms of computing. Data is basically information that can be translated into a particular form for efficient movement and processing.

What is a Database?

The database is an organized collection of structured data to make it easily accessible, manageable and update. In simple words, you can say, a database in a place where the data is stored. The best analogy is the library. The library contains a huge collection of books of different genres, here the library is database and books are the data.

Database Components

The major components of the Database are:

- **Hardware**

This consists of a set of physical electronic devices such as I/O devices, storage devices and many more. It also provides an interface between computers and real-world systems.

- **Software**

This is the set of programs that are used to control and manage the overall Database. It also includes the DBMS software itself. The Operating System, the network software being used to share the data among the users, the application programs used to access data in the DBMS.

- **Data**

Database Management System collects, stores, processes, and accesses data. The Database holds both the actual or operational data and the metadata.

Types of Database

There are a few types that are very important and popular.

- Relational Database
- Object-Oriented Database
- Distributed Database
- Graph Database
- Cloud Database
- Centralization Database

What is SQL?

Structured Query language SQL is pronounced as "S-Q-L" or sometimes as "See-Quel" which is the standard language for dealing with **Relational Databases**.

It is effectively used to insert, search, update, delete, modify database records. It doesn't mean SQL cannot do things beyond that. In fact, it can do a lot more other things as well. SQL is regularly used not only by database administrators but also by the developers to write data integration scripts and data analysts.

Advantages

1. Reduced data redundancy.
2. Easier data integrity from application programs
3. Data security is also improved.

Database Connection:

There are the following steps to connect a python application to our database.

1. Import pymysql module
2. Create the connection object.
3. Create the cursor object
4. Execute the query

Creating the connection

To create a connection between the MySQL database and the python application, the **connect()** method of pymysql module is used. Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

The syntax to use the connect() is given below.

Connection-Object= pymysql.connect (host = <host-name> , user = <username> , passwd = <password>, database=<name>)

Example

```
import pymysql as pysq
#Create the connection object
myconn = pysq.connect(host = "localhost", user = "root",passwd = "", d
atabase = "mydb")
#printing the connection object
print(myconn)
```

Creating a cursor object:

We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.
The syntax to create the cursor object is given below.

<my_cur> = conn.cursor()

Using the methods of it you can execute SQL statements, fetch data from the result sets, call procedures.

Example

```
import pymysql as pysq

#establishing the connection
conn = pysq.connect(
   user='root', password='password', host='127.0.0.1', database='mydb'
)
#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

Creating the table

We can create the new table by using the CREATE TABLE statement of SQL. In our database PythonDB, the table Employee will have the four columns, i.e., name, id, salary, and department_id initially.

```
import pymysql as pysq

#Create the connection object
myconn = pysq.connect(host = "localhost", user = "root",passwd = "",da
tabase = "PythonDB")

#creating the cursor object
cur = myconn.cursor()

try:
    #Creating a table with name Employee having four columns i.e., nam
e, id, salary, and department id
    dbs = cur.execute("create table Employee(name varchar(20) not null
, id int(20) not null primary key, salary float not null, Dept_id int
not null)")
except:
  myconn.rollback()
  myconn.close()
```

Adding a record to the table:

The **INSERT INTO** statement is used to add a record to the table. In python, we can mention the format **specifier (%s)** in place of values.

We provide the actual values in the form of tuple in the execute() method of the cursor.

```
import pymysql as pysq

#Create the connection object
myconn = pysq .connect(host = "localhost", user = "root",passwd = "",d
atabase = "mydb")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) va
lues (%s, %s, %s, %s, %s)"

#The row values are provided in the form of tuple
val = ("Amar", 110, 25000.00, 201, "Latur")

try:
    #inserting the values into the table
    cur.execute(sql,val)

    #commit the transaction
    myconn.commit()

except:
    myconn.rollback()

print(cur.rowcount,"record inserted!")
myconn.close()
```

Insert multiple rows:

We can also insert multiple rows at once using the python script. The multiple rows are mentioned as the list of various tuples.

Each element of the list is treated as one particular row, whereas each element of the tuple is treated as one particular column value (attribute).

```
import pymysql as pysql

#Create the connection object
myconn = pysql.connect(host = "localhost", user = "root",passwd = "",d
atabase = "mydb")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) va
lues (%s, %s, %s, %s, %s)"
val = [("Amar", 102, 25000.00, 201, "Pune"),("Ram",103,25000.00,202,"M
umbai"),("Shubham",104,90000.00,201,"Latur")]

try:
    #inserting the values into the table
    cur.executemany(sql,val)

    #commit the transaction
    myconn.commit()
    print(cur.rowcount,"records inserted!")

except:
    myconn.rollback()
    myconn.close()
```

Read Operation:

The SELECT statement is used to read the values from the databases. We can restrict the output of a select query by using various clause in SQL like where, limit, etc.

Python provides the **fetchall()** method returns the data stored inside the table in the form of rows. We can iterate the result to get the individual rows.

```
import pymysql as pysql

#Create the connection object
myconn = pysql.connect(host = "localhost", user = "root",passwd = "",d
atabase = "mydb")

#creating the cursor object
cur = myconn.cursor()

try:
    #Reading the Employee data
    cur.execute("select * from Employee")

    #fetching the rows from the cursor object
    result = cur.fetchall()

   #printing the result
      for x in result:
        print(x);
except:
    myconn.rollback()
    myconn.close()
```

Formatting the result:

We can format the result by iterating over the result produced by the fetchall() or fetchone() method of cursor object since the result exists as the tuple object which is not readable.

```
import pymysql as pysql

#Create the connection object
myconn = pysql.connect(host = "localhost", user = "root",passwd = "",d
atabase = "mydb")

#creating the cursor object
cur = myconn.cursor()

try:

    #Reading the Employee data
    cur.execute("select name, id, salary from Employee")

    #fetching the rows from the cursor object
    result = cur.fetchall()

    print("Name    id    Salary");
    for row in result:
        print("%s    %d    %d"%(row[0],row[1],row[2]))
except:
    myconn.rollback()
    myconn.close()
```

Update Operation

The UPDATE-SET statement is used to update any column inside the table. The following SQL query is used to update a column.

> update Employee set name = ‘ABC’ where id = 100

```
import pymysql as pysq

#Create the connection object
myconn = pysq.connect(host = "localhost", user = "root",passwd = "",da
tabase = "PythonDB")

#creating the cursor object
cur = myconn.cursor()

try:
    #updating the name of the employee whose id is 110
    cur.execute("update Employee set name = 'ABC' where id = 10")
    myconn.commit()
except:

    myconn.rollback()

myconn.close()
```

Delete Operation

The DELETE FROM statement is used to delete a specific record from the table. Here, we must impose a condition using WHERE clause otherwise all the records from the table will be removed.

The following SQL query is used to delete the employee detail whose id is 110 from the table.

```
delete from Employee where id = 110
```

```
import pymysql as pysq

#Create the connection object
myconn = pysq.connect(host = "localhost", user = "root",passwd = "",da
tabase = "PythonDB")

#creating the cursor object
cur = myconn.cursor()

try:
    #Deleting the employee details whose id is 110
    cur.execute("delete from Employee where id = 110")
    myconn.commit()
except:

    myconn.rollback()
    myconn.close()
```