

Database Operations using MySql in Python

What is Data?

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

What is Database?

A **database** is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.



RDBMS (Relational Database Management System)

The word RDBMS is termed as 'Relational Database Management System.' It is represented as a table that contains rows and column.

RDBMS is based on the Relational model; it was introduced by E. F. Codd.

A relational database contains the following components:

- Table
- Record/ Tuple
- Field/Column name /Attribute
- Instance
- Schema
- Keys

An RDBMS is a tabular DBMS that maintains the security, integrity, accuracy, and consistency of the data.

1. Steps to create database:

Create database <dbname>

2. Enter the database: use database
3. Create table:

```
CREATE TABLE People(  
  id int NOT NULL AUTO_INCREMENT,  
  name varchar(45) NOT NULL,  
  occupation varchar(35) NOT NULL,  
  age int,  
  PRIMARY KEY (id)  
);
```

4. Insert records in a table:

INSERT INTO People (id, **name**, occupation, age)

VALUES (101, 'Peter', 'Engineer', 32);

(or)

INSERT INTO People **VALUES**

(102, 'Joseph', 'Developer', 30),

(103, 'Mike', 'Leader', 28),

(104, 'Stephen', 'Scientist', 45);

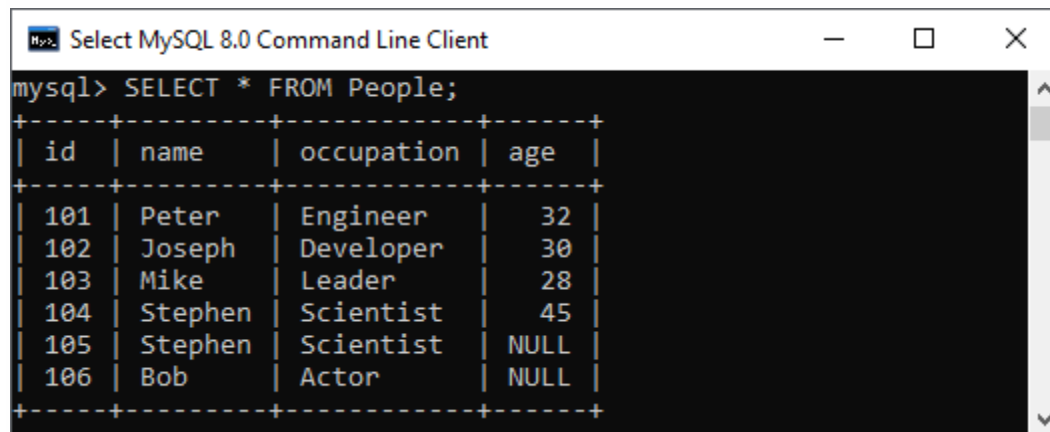
(or)

INSERT INTO People (**name**, occupation)

VALUES ('Stephen', 'Scientist'), ('Bob', 'Actor');

5. To see records of the table:

SELECT * **FROM** People;



The screenshot shows a terminal window titled "Select MySQL 8.0 Command Line Client". The prompt is "mysql>". The command entered is "SELECT * FROM People;". The output is a table with 6 rows and 4 columns: id, name, occupation, and age. The data is as follows:

id	name	occupation	age
101	Peter	Engineer	32
102	Joseph	Developer	30
103	Mike	Leader	28
104	Stephen	Scientist	45
105	Stephen	Scientist	NULL
106	Bob	Actor	NULL

In Python, We can use the following modules to communicate with MySQL.

- MySQL Connector Python
- PyMySQL
- MySQLDB
- MySqIClient
- OurSQL

You can choose any of the above modules as per your requirements. The way of accessing the MySQL database remains the same. I recommend you to use any of the following two modules:-

1. **MySQL Connector Python**
2. **PyMySQL**

Advantages and benefits of MySQL Connector Python: –

- MySQL Connector Python is written in pure Python, and it is self-sufficient to execute database queries through Python.
- It is an official Oracle-supported driver to work with MySQL and Python.
- It is Python 3 compatible, actively maintained.

How to connect MySQL database in Python?

Arguments required to connect

You need to know the following detail of the MySQL server to perform the connection from Python.

Argument	Description
Username	The username that you use to work with MySQL Server. The default username for the MySQL database is a root .

Argument	Description
Password	Password is given by the user at the time of installing the MySQL server. If you are using root then you won't need the password.
Host name	The server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost or its IP 127.0.0.0
Database name	The name of the database to which you want to connect and perform the operations.

How to Connect to MySQL Database in Python

1. Install MySQL connector module

Use the pip command to install MySQL_connector_Python.
`pip install mysql-connector-python`

2. Import MySQL connector module

Import using a `import mysql.connector` statement so you can use this module's methods to communicate with the MySQL database.

3. Use the connect() method

Use the `connect()` method of the MySQL Connector class with the required arguments to connect MySQL. It would return a `MySQLConnection` object if the connection established successfully

4. Use the cursor() method

Use the `cursor()` method of a `MySQLConnection` object to create a cursor object to perform various SQL operations.

5. Use the execute() method

The execute() methods run the SQL query and return the result.

6. Extract result using fetchall()

Use `cursor.fetchall()` or `fetchone()` or `fetchmany()` to read query result.

7. Close cursor and connection objects

use `cursor.close()` and `connection.close()` method to close open connections after your work completes

.....

Program to connect to mysql using 'pymysql' module:

Example 1:

```
...
    1. Download the pymysql module
    2. Go to your command prompt
    3. Type the following command: pip install pymysql
    4. Module will be downloaded and installed
...
# Prog to connect to mysql database using the module 'pymysql'

import pymysql

# connect() fun to get connected to mysql db
con = pymysql.connect(host='localhost', user='root', password='root',
db='mydb1')
print("Connection success...")

# create a cursor to access the table
cur = con.cursor()
print("Cursor create success")

# prepare the query
query = "select * from student"

# execute the query
cur.execute(query)
print("query execution success")
```

```

# get the data to a variable
# function fetchall() gets the resultset(executed query rows)
data = cur.fetchall()

# print the resultset
for i in data:
    print(i)

```

Program to connect to MySql Database using module

'mysql-connector-python':

Example 2:

```

import mysql.connector
from mysql.connector import Error

try:
    connection = mysql.connector.connect(host='localhost',
                                         database='mydb1',
                                         user='root',
                                         password='root')

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record)

except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")

```

Create MySQL table from Python:

Example 3:

```
import mysql.connector

try:
    connection = mysql.connector.connect(host='localhost',
                                         database='mydb1',
                                         user='root',
                                         password='root')

    mySql_Create_Table_Query = """CREATE TABLE Laptop (
                                   Id int(11) NOT NULL,
                                   Name varchar(250) NOT NULL,
                                   Price float NOT NULL,
                                   Purchase_date Date NOT NULL,
                                   PRIMARY KEY (Id)) """

    cursor = connection.cursor()
    result = cursor.execute(mySql_Create_Table_Query)
    print("Laptop Table created successfully ")

except mysql.connector.Error as error:
    print("Failed to create table in MySQL: {}".format(error))
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

Use Python Variables in a MySQL Insert Query

Example 4:

```
import mysql.connector

def insert_variables_into_table(id, name, price, purchase_date):
    try:
        connection = mysql.connector.connect(host='localhost',
```



```

mySql_insert_query = """INSERT INTO Laptop (Id, Name, Price,
Purchase_date)
                        VALUES (%s, %s, %s, %s) """

cursor = connection.cursor()
current_Date = datetime.now()
# convert date in the format you want
formatted_date = current_Date.strftime('%Y-%m-%d %H:%M:%S')
insert_tuple = (7, 'Acer Predator Triton', 2435, current_Date)

result = cursor.execute(mySql_insert_query, insert_tuple)
connection.commit()
print("Date Record inserted successfully")

except mysql.connector.Error as error:
    connection.rollback()
    print("Failed to insert into MySQL table {}".format(error))

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")

```

.....

Database Transaction

The database transaction represents a **single unit of work**. Any operation which modifies the state of the MySQL database is a transaction. Let see in detail what is database transaction. For example, take a sample of a Bank amount transfer, which involves two significant transactions.

- Withdrawal of money from account A
- Deposit Money to Account B

If the first Transaction is executed successfully but the second failed, in this case, we need to re-deposit money back to account A. To manage such instances, we need transaction management.

ACID properties:

We can study transaction management well. ACID stands for Atomicity, Consistency, isolation, and durability.

- **Atomicity:** means all or nothing. Either all transactions are successful or none. You can group SQL statements as one logical unit, and if any query fails, the whole transaction fails.
- **Consistency:** It ensures that the database remains in a consistent state after performing a transaction.
- **Isolation:** It ensures that the transaction is isolated from other transactions.
- **Durability:** It means once a transaction has been committed, it persists in the database irrespective of power loss, error, or restart system.

Python MySQL Commit(), rollback() and setAutoCommit() to manage transactions:

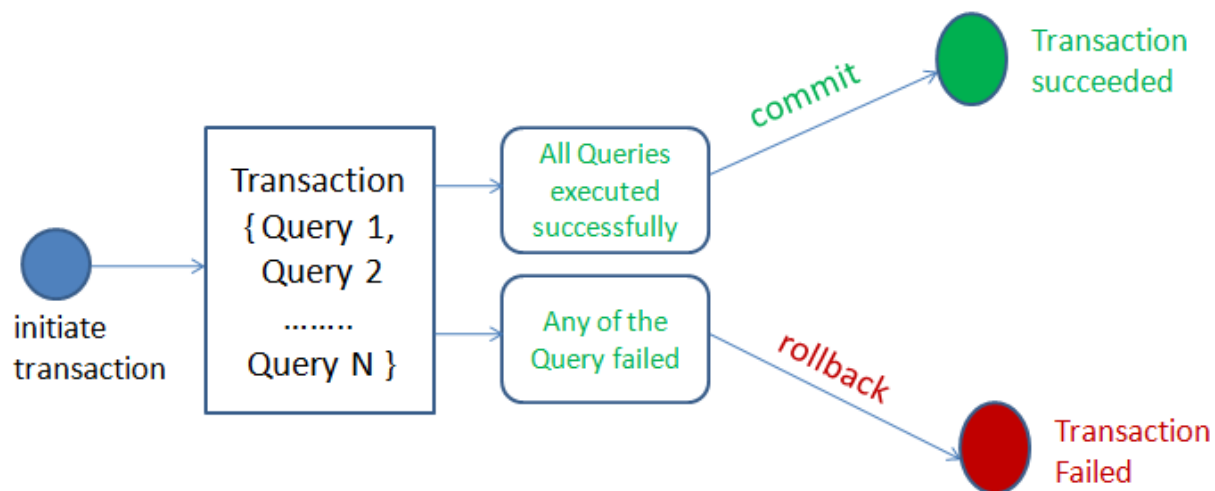
Please follow the below steps to manage MySQL transactions in Python: –

- Create MySQL database connections in Python.
- Prepare the SQL queries that you want to run as a part of a transaction. For example, we can combine two SQL queries (withdrawal money and deposit money query) in a single transaction.
- Set an auto-commit property of MySQL connection to false.
- Execute all queries one by one using the cursor.execute()
- If all queries execute successfully, commit the changes to the database
- If one of the queries failed to execute, then rollback all the changes.
- Catch any SQL exceptions that may occur during this process
- Close the cursor object and MySQL database connection

Methods to manage MySQL Database Transactions in Python

Python MySQL Connector provides the following method to manage database transactions.

- `commit(): MySQLConnection.commit()` method sends a COMMIT statement to the MySQL server, committing the current transaction. After the successful execution of a query make changes persistent into a database using the `commit()` of a connection class.
- `rollback(): MySQLConnection.rollback` revert the changes made by the current transaction. When one of the transactions fails to execute, and you want to revert or undo all your changes, call a rollback method of MySQL connection object.
- `autoCommit() : MySQLConnection.autocommit` value can be as True or False to enable or disable the auto-commit feature of MySQL. By default, its value is False.



Python example to manage MySQL transactions using commit and rollback:

Note: Create 'account_A' and 'account_B' and insert records into these tables before executing this program.

Example 7:

```
import mysql.connector

try:
    conn = mysql.connector.connect(host='localhost',
                                   database='mydb1',
                                   user='root',
                                   password='root')

    conn.autocommit = False
    cursor = conn.cursor()
    # withdraw from account A
    sql_update_query = """Update account_A set balance = 1000 where id
= 1"""
    cursor.execute(sql_update_query)

    # Deposit to account B
    sql_update_query = """Update account_B set balance = 1500 where id
= 2"""
    cursor.execute(sql_update_query)
    print("Record Updated successfully ")

    # Commit your changes
    conn.commit()

except mysql.connector.Error as error:
    print("Failed to update record to database rollback:
{}".format(error))
    # reverting changes because of exception
    conn.rollback()
finally:
    # closing database connection.
    if conn.is_connected():
        cursor.close()
        conn.close()
        print("connection is closed")
```