# MYSQL CONNECTIVITY NOTES

**Steps for creating Database Connectivity Applications**

- **Start Python**
- **Import the packages required for database programming-** **import mysql.connector**
- **Open a connection to database-** **conn=m.connect(…..)**
- **Create a cursor instance ---** **cur=conn.cursor()**
- **Execute a query ----** **cur.execute("….ddl/dml commands)**
- **Extract data from result set----** **cur.fetchall/fetchone/fetchmany--**
- **Clean up the environment----** **conn.close()**

---

**For database interface/database programming ,connection must be established.**
**Before establishing connection there must be mysql installed on the system and a database and table is already created.**
**In following way we can establish a connection with mysql database through mysql.connector.**

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="db1")
print(mydb)
```

OR

```
import mysql.connector as m
con=m.connect(host="localhost",user="root",passwd="1234")
if con.is_connected():
        print("successfully connected")
```

---

**Cursor object :**

The MySQLCursor class instantiates objects that can execute operations such as SQL statements.
Cursor objects interact with the MySQL server using a MySQLConnection object.

```
import mysql.connector as m
# Open database connection
db =
m.connect(host="localhost",user="root",passwd="1234")

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("show databases")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print ("Database version : %s " % data)

# disconnect from server
db.close()
```

OR

```
import mysql.connector as m
# Open database connection
db =
m.connect(host="localhost",user="root",passwd="1234")

cursor = db.cursor()

cursor.execute("show databases")
# Fetch all rows using fetchall() method.
data = cursor.fetchall()
print ("Database version : %s " % data)
db.close()
```

OR

```
data = cursor.fetchall()
for i in data:
        print(i)
db.close()
```

To fetch some useful information from the database, can use either fetchone() method to fetch single record or **fetchall()** method to fetch multiple values from a database table.
**fetchone()** − It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.
**fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.
**rowcount** − This is a read-only attribute and returns the number of rows that were affected by an execute() method

## Creating Database Table

Once a database connection is established, we are ready to create tables or records into the database tables using **execute** method of the created cursor.

## Example

To create Database table EMPLOYEE −

```
import mysql.connector
```

```
import mysql.connector
```

```
db =
mysql.connector.connect(host="localhost",user="root",
passwd="1234",database="ss")
cursor = db.cursor()
# Drop table if it already exist using
execute() method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")
sql = """CREATE TABLE EMPLOYEE (
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME  CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT )"""
cursor.execute(sql)
db.close()
```

OR

```
db =
mysql.connector.connect(host="localhost",user="root",
passwd="1234",database="ss")
cursor = db.cursor()
# Drop table if it already exist using execute()
method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")
cursor.execute("CREATE TABLE EMPLOYEE (FIRST_NAME
CHAR(20) NOT NULL,LAST_NAME  CHAR(20),AGE INT,  SEX
CHAR(1),INCOME FLOAT )")
cursor.execute("describe employee")
for I in cursor:
      print(I)
cursor.execute('show tables')
for I in cursor:
      print(I)
db.close()
```

**INSERT Operation**-to create records into a database table.

**Example**- *INSERT* statement to create a record into EMPLOYEE table –

```
import mysql.connector
db =
mysql.connector.connect(host="localhost",user="root",passwd="1234",d
atabase="ss")
cursor = db.cursor()
# Prepare SQL query to INSERT a record into the
database.
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
         LAST_NAME, AGE, SEX, INCOME)
         VALUES ('Mac', 'Mohan', 20, 'M',
2000)"""
try:
   cursor.execute(sql)
   db.commit()
except:
   db.rollback()
db.close()
```

OR

```
import mysql.connector
db =
mysql.connector.connect(host="localhost",user
="root",passwd="1234",database="ss")
cursor = db.cursor()
# Prepare SQL query to INSERT a
record into the database.
cursor.execute("INSERT INTO
EMPLOYEE(FIRST_NAME,LAST_NAME, AGE,
SEX, INCOME)VALUES ('Mac', 'Mohan',
20, 'M', 2000)")
db.commit()
db.close()
```

---

**Manage Database Transaction**

Database transaction represents a single unit of work. Any operation which modifies the state of the MySQL database is a transaction.

**commit** – MySQLConnection.commit() method sends a COMMIT statement to the MySQL server, committing(save) the current transaction.(commit() always use with insert, alter, delete, update, delete commands)

**rollback** – MySQLConnection.rollback revert(undo) the changes made by the current transaction.

---

Example Q:-All the records from EMPLOYEE table having salary more than 1000 –

```
import mysql.connector
db =
mysql.connector.connect(host="localhost",user="root",passwd="1234",data
base="ss")
cursor = db.cursor()
sql = "SELECT * FROM EMPLOYEE WHERE INCOME > 1000"
try:
   cursor.execute(sql)
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
print("fname=%s,lname=%s,age=%d,sex=%s,income=%d"%(fname,
lname, age, sex, income )
except:
   print ("Error: unable to fetch data")
db.close()
```

OR

```
import mysql.connector
db =
mysql.connector.connect(host="localhost",user="roo
t",passwd="1234",database="ss")
cursor = db.cursor()
cursor.execute("SELECT * FROM
EMPLOYEE WHERE INCOME > 1000")
d=cursor.fetchall()
for i in d:
     print(i[0],i[1],i[2],i[3],i[4])
db.close()
          OR
for i in d:
     print(i)
db.close()
```