- What is Data bases : A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).
- Who uses a database? Your grocery store, bank, restaurant, online shopping sites, hospital, favorite clothing store and mobile service provider, for instance all use databases to keep track of customer, inventory, employee and accounting information.

here we will ducuss mysql databases using python programming.

Steps to install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

- for using mysql,we will use the driver "MySQL Connector".
- We recommend that you use PIP to install "MySQL Connector".
- PIP is most likely already installed in your Python environment.
- Navigate your command line to the location of PIP, and type the following:

commands to install necessary mysql connector :

- install mysql connector pkg using cmd: *pip install mysql-connector-python*
- use in terminal: *python -m pip install mysql-connector-python*

import module

- Test MySQL Connector: To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content:

```
pip install mysql-connector-python

Requirement already satisfied: mysql-connector-python in c:\users\
rajen\anaconda3\lib\site-packages (8.3.0)
Note: you may need to restart the kernel to use updated packages.

import mysql.connector
```

NOTE : if we'll not get error using above commands than, it means we success installed mysql in our python environments.

Create Database

PYTHON SYNTAX for mysql :

SYNTAX : import mysql.connector

database_object = mysql.connector.connect(host = "localhost", user = "root", passwd = "PASSWORD", database = "DATABASE_NAME")

mycursor = mydb.cursor()

mycursor.execute("*********MYSQL QUERIES**********")

```
statements/code......
```

VARIABLE USED IN SYNTAX :

- database_object : It is a casual object to store the values and allow to use database in programming
- HOST_NAME : The MySQL hostname defines the location of your MySQL server and database.
- PASSWORD : by default, there is no password but if you create then put it there.
- DATABASE_NAME : Define a Database name that you want to create or want to work on existing any database.
- MYSQL_QUERIES : all queries same as used in mysql.

NOTE : Default user for MySQL is "root", and server "localhost" with no password on database.

Create Connection:

Start by creating a connection to the database. Use the username and password from your MySQL database:

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000"
)
print(mydb)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at
0x0000021245279210>
```

Creating a Database:

To create a database in MySQL, use the "CREATE DATABASE" statement: Let's create a database named 'mydatabase'

Note : If the above code was executed with no errors, you have successfully created a database.

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000'
```

```
)

mycusor = mydb.cursor()

mycusor.execute("CREATE DATABASE mydatabase")
```

ERROR : you'll get following error if you tried to create databases who are already exists in database.

Output :

MySQLInterfaceError Traceback (most recent call last) File ~\anaconda3\Lib\site-packages\mysql\connector\connection_cext.py:661, in CMySQLConnection.cmd_query(self, query, raw, buffered, raw_as_string) 660 query = query.encode("utf-8") --> 661 self._cmysql.query(662 query, 663 raw=raw, 664 buffered=buffered, 665 raw_as_string=raw_as_string, 666 query_attrs=self.query_attrs, 667) 668 except MySQLInterfaceError as err:

MySQLInterfaceError: Can't create database 'mydatabase'; database exists

The above exception was the direct cause of the following exception:

DatabaseError Traceback (most recent call last) Cell In[11], line 10 3 mydb = mysql.connector.connect(4 host = 'localhost', 5 user = 'root', 6 passwd = 'EHARAmaika@2000') 8 mycursor = mydb.cursor() ---> 10 mycursor.execute("CREATE DATABASE mydatabase")

File ~\anaconda3\Lib\site-packages\mysql\connector\cursor_cext.py:374, in CMySQLCursor.execute(self, operation, params, multi) 369 raise ProgrammingError(370 "Not all parameters were used in the SQL statement" 371) 373 try: --> 374 result = self._cnx.cmd_query(375 stmt, 376 raw=self._raw, 377 buffered=self._buffered, 378 raw_as_string=self._raw_as_string, 379) 380 except MySQLInterfaceError as err: 381 raise get_mysql_exception(382 msg=err.msg, errno=err.errno, sqlstate=err.sqlstate 383) from err

File ~\anaconda3\Lib\site-packages\mysql\connector\opentelemetry\context_propagation.py:74, in with_context_propagation..wrapper(cnx, *args, **kwargs) 72 """Context propagation decorator.""" 73 if not OTEL_ENABLED or not cnx.otel_context_propagation: ---> 74 return method(cnx, *args, **kwargs) 76 current_span = trace.get_current_span() 77 tp_header = None

File ~\anaconda3\Lib\site-packages\mysql\connector\connection_cext.py:669, in CMySQLConnection.cmd_query(self, query, raw, buffered, raw_as_string) 661 self._cmysql.query(662 query, 663 raw=raw, (...) 666 query_attrs=self.query_attrs, 667) 668 except MySQLInterfaceError as err: --> 669 raise get_mysql_exception(670 err.errno, msg=err.msg, sqlstate=err.sqlstate 671) from err 672 except AttributeError as err: 673 addr = (674 self._unix_socket if self._unix_socket else f"{self._host}:{self._port}" 675)

DatabaseError: 1007 (HY000): Can't create database 'mydatabase'; database exists

Check all Exists Databases:

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement:

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase'
)

mycursor = mydb.cursor()
mycursor.execute("show databases")

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

```
('information_schema',)
('mydatabase',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
('world',)
```

ERROR : If the database does not exist, you will get an error.

Output :

MySQLInterfaceError Traceback (most recent call last) ~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in _open_connection(self) 287 try: --> 288 self._cmysql.connect(**cnx_kwargs) 289 self._cmysql.converter_str_fallback = self._converter_str_fallback

MySQLInterfaceError: Unknown database 'mybase'

The above exception was the direct cause of the following exception:

ProgrammingError Traceback (most recent call last) ~\AppData\Local\Temp\ipykernel_4760\2675299089.py in 1 import mysql.connector 2 ----> 3 mydb = mysql.connector.connect(host = 'localhost', 4 user = 'root', 5 passwd = 'EHARAmaika@2000',

~\anaconda3\lib\site-packages\mysql\connector\pooling.py in connect(*args, **kwargs) 291 292 if CMySQLConnection and not use_pure: --> 293 return CMySQLConnection(*args, **kwargs) 294 return MySQLConnection(*args, **kwargs) 295

~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in **init**(self, **kwargs) 116 **117 if kwargs: --> 118 self.connect(**kwargs) 119 120 def _add_default_conn_attrs(self) -> None:

~\anaconda3\lib\site-packages\mysql\connector\abstracts.py in connect(self, **kwargs) 1176 1177 self.disconnect() -> 1178 self._open_connection() 1179 # Server does not allow to run any other statement different from ALTER 1180 # when user's password has been expired.

~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in _open_connection(self) 291 self.converter.str_fallback = self._converter_str_fallback 292 except MySQLInterfaceError as err: --> 293 raise get_mysql_exception(294 msg=err.msg, errno=err.errno, sqlstate=err.sqlstate 295) from err

ProgrammingError: 1049 (42000): Unknown database 'mybase'

Create Table :

Creating a Table:

To create a table in MySQL, use the "CREATE TABLE" statement. Make sure you define the name of the database when you create the connection. create a table named 'customer'

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase'
)

mycursor = mydb.cursor()

mycursor.execute('CREATE TABLE customer(id int(3), name varchar(10), place varchar(10))')
```

NOTE : If the above code was executed with no errors, you have now successfully created a table.

- Let's create one more table

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase'
)

mycursor = mydb.cursor()

mycursor.execute('CREATE TABLE mycustomer(name varchar(15), place varchar(15))')
```

Check all Exists tables:

You can check if a table exist by listing all tables in your database with the "SHOW TABLES" statement:

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase'
)

mycursor = mydb.cursor()
mycursor.execute("show tables")

for x in mycursor:
    print(x)

('customer',)
('mycustomer',)
```

Primary Key:

- When creating a table, you should also create a column with a unique key for each record.
- This can be done by defining a PRIMARY KEY.
- We use the statement "INT AUTO_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers(ID INT AUTO_INCREMENT, NAME
VARCHAR(255), ADDRESS VARCHAR(255), PRIMARY KEY(ID))")
```

- Let's see it with ccreating one more table

```
import mysql.connector

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

```
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE user(FNAME VARCHAR(255), LNAME
VARCHAR(255), ADDRESS VARCHAR(255))")
```

add primary key in existing table:

- If the table already exists, use the ALTER TABLE keyword:
- Create primary key on an existing table:

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase'
)

mycursor = mydb.cursor()

mycursor.execute('ALTER TABLE user ADD COLUMN ID INT PRIMARY KEY')
```

insert table :

Insert single values in table:

- To fill a table in MySQL, use the "INSERT INTO" statement.
- Insert a record in the "customers" table:(single or multiple values)

Insert single values in table:

- To fill a table in MySQL, use the "INSERT INTO" statement.
- Insert a record in the "customers" table:(single value)

```
import mysql.connector

# Create a connection to the MySQL database
mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

# Create a cursor object
mycursor = mydb.cursor()
```

```
# Execute an INSERT statement
sql = "insert into customers(NAME, ADDRESS) values (%s, %s)"
val = ("JOHN", "HIGHWAY 21")

mycursor.execute(sql, val)

# Commit the changes
mydb.commit()

# Close the connection
print(mycursor.rowcount,"record inserted.")
```

```
1 record inserted.
```

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Insert multiple values in table:

- To fill a table in MySQL, use the "INSERT INTO" statement.
- Insert a record in the "customers" table:(multiple values)

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

sql = "insert into customers(NAME, ADDRESS) values (%s, %s)"
val = [('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')]

mycursor.executemany(sql,val)
mydb.commit()
```

```
print(mycursor.rowcount," rows was intserted.")
```

```
13  rows was intserted.
```

- **Let's see with one more table** mycustomer.

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

sql = "insert into mycustomer(name, place) values (%s, %s)"
val = [('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount,"rows was inserted.")
```

```
13 rows was inserted.
```

- **Let's see with one more table** user.

Get Inserted ID of last value:

You can get the id of the row you just inserted by asking the cursor object. Note : If you insert more than one row, the id of the last inserted row is returned.

```
import mysql.connector

mydb = mysql.connector.connect(
```

```
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

sql = "insert into customers(NAME, ADDRESS) values (%s, %s)"
val = ("Michelle", "Blue Village")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount,"record inserted, last recorded
ID :",mycursor.lastrowid)
```

```
1 record inserted, last recorded ID : 15
```

select table :

Select data From a Table:

To select from a table in MySQL, use the "SELECT" statement:

1. fetchall() : The fetchall() method in Python is used to fetch all (or all remaining) rows of a query result set and returns a list of tuples. If no more rows are available, it returns an empty list.

Select all records from the "customers" table, and display the result:

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    passwd = "EHARAmaika@2000",
    database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

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

```
(1, 'JOHN', 'HIGHWAY 21')
(2, 'Peter', 'Lowstreet 4')
```

```
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
(8, 'Richard', 'Sky st 331')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(11, 'Ben', 'Park Lane 38')
(12, 'William', 'Central st 954')
(13, 'Chuck', 'Main Road 989')
(14, 'Viola', 'Sideway 1633')
(15, 'Michelle', 'Blue Village')
```

Note: We use the fetchall() method, which fetches all rows from the last executed statement.

- you can print result with specific columms.

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("SELECT NAME, ADDRESS FROM customers")

myresult = mycursor.fetchall()

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

```
('JOHN', 'HIGHWAY 21')
('Peter', 'Lowstreet 4')
('Amy', 'Apple st 652')
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
('Betty', 'Green Grass 1')
('Richard', 'Sky st 331')
('Susan', 'One way 98')
('Vicky', 'Yellow Garden 2')
('Ben', 'Park Lane 38')
('William', 'Central st 954')
('Chuck', 'Main Road 989')
('Viola', 'Sideway 1633')
('Michelle', 'Blue Village')
```

```
import mysql.connector

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

```
                                user = "root",
                                passwd = "EHARAmaika@2000",
                                database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("SELECT ADDRESS FROM customers")

myresult = mycursor.fetchall()

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

```
('HIGHWAY 21',)
('Lowstreet 4',)
('Apple st 652',)
('Mountain 21',)
('Valley 345',)
('Ocean blvd 2',)
('Green Grass 1',)
('Sky st 331',)
('One way 98',)
('Yellow Garden 2',)
('Park Lane 38',)
('Central st 954',)
('Main Road 989',)
('Sideway 1633',)
('Blue Village',)
```

fetchone() : If you are only interested in one row, you can use the fetchone() method. The fetchone() method will return the first row of the result: Fetch only one row:

Select first record from the "customers" table, and display the result:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost",
                                user = "root",
                                passwd = "EHARAmaika@2000",
                                database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

```
(1, 'JOHN', 'HIGHWAY 21')
```

Select With a Filter:

- select with where: When selecting records from a table, you can filter the selection by using the "WHERE" statement:

Select and print record(s) where the address is "Park Lane 38" in table customers :

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers WHERE ADDRESS = 'Park Lane
38'")

myresult = mycursor.fetchall()

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

```
(11, 'Ben', 'Park Lane 38')
```

select and print record(s) with specific in records.

- Wildcard Characters:
  You can also select the records that starts, includes, or ends with a given letter or phrase. Use the % to represent wildcard characters:

Select and print record(s) where the address contains the word "way" in table customers:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE ADDRESS LIKE '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
(1, 'JOHN', 'HIGHWAY 21')
(9, 'Susan', 'One way 98')
(14, 'Viola', 'Sideway 1633')
```

- ## Prevent SQL Injection:
  When query values are provided by the user, you should escape the values. This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database. The mysql.connector module has methods to escape query values:

Escape query values by using the placholder %s method:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE ADDRESS = %s"
add = ("Yellow Garden 2",)

mycursor.execute(sql,add)

myresult = mycursor.fetchall()

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

```
(10, 'Vicky', 'Yellow Garden 2')
```

Python MySQL Order By:

Sort the Result: Use the ORDER BY statement to sort the result in ascending or descending order. The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

Sort the result alphabetically by name: result:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = 'SELECT * FROM customers ORDER BY name'

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
(3, 'Amy', 'Apple st 652')
(11, 'Ben', 'Park Lane 38')
(7, 'Betty', 'Green Grass 1')
(13, 'Chuck', 'Main Road 989')
(4, 'Hannah', 'Mountain 21')
(1, 'JOHN', 'HIGHWAY 21')
(5, 'Michael', 'Valley 345')
(15, 'Michelle', 'Blue Village')
(2, 'Peter', 'Lowstreet 4')
(8, 'Richard', 'Sky st 331')
(6, 'Sandy', 'Ocean blvd 2')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(14, 'Viola', 'Sideway 1633')
(12, 'William', 'Central st 954')
```

Sort by ORDER in DESC: Use the DESC keyword to sort the result in a descending order.

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = 'SELECT * FROM customers ORDER BY name DESC'

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
(12, 'William', 'Central st 954')
(14, 'Viola', 'Sideway 1633')
(10, 'Vicky', 'Yellow Garden 2')
(9, 'Susan', 'One way 98')
(6, 'Sandy', 'Ocean blvd 2')
(8, 'Richard', 'Sky st 331')
(2, 'Peter', 'Lowstreet 4')
(15, 'Michelle', 'Blue Village')
(5, 'Michael', 'Valley 345')
(1, 'JOHN', 'HIGHWAY 21')
(4, 'Hannah', 'Mountain 21')
(13, 'Chuck', 'Main Road 989')
(7, 'Betty', 'Green Grass 1')
(11, 'Ben', 'Park Lane 38')
(3, 'Amy', 'Apple st 652')
```

delete table :

Delete Record

You can delete records from an existing table by using the "DELETE FROM" statement:

Delete any record where the address is "Mountain 21":

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE ADDRESS = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount,"record(s) deleted.")
```

```
1 record(s) deleted.
```

NOTE :

- Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.
- the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in delete statements. This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database. The mysql.connector module uses the placeholder %s to escape values in the delete statement:

Escape values by using the placeholder %s method:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )
```

```
mycursor.execute(sql, adr)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

```
1 record(s) deleted
```

Drop table

Drop a Table

You can delete an existing table by using the "DROP TABLE" statement:

Delete the table "customers":

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)
```

Drop Only if Exist

If the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error.

Delete the table "customers" if it exists:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)
```

Update table

Update the records in Table

You can update existing records in a table by using the "UPDATE" statement:

Overwrite the address column from "Valley 345" to "Canyon 123":

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "UPDATE mycustomer SET place = 'Canyon 123' WHERE place =
'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

```
1 record(s) affected
```

NOTE :

- Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.
- Notice the WHERE clause in the UPDATE syntax:*** The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in update statements. This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database. The mysql.connector module uses the placeholder %s to escape values in the delete statement:

Escape values by using the placeholder %s method:

Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in update statements. This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database. The mysql.connector module uses the placeholder %s to escape values in the delete statement:

Escape values by using the placeholder %s method:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "UPDATE mycustomer SET place = %s WHERE place = %s"
val =('Valley 345','Canyon 123')

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

```
1 record(s) affected
```

Limit the Result

You can limit the number of records returned from the query, by using the "LIMIT" statement:

Select the 5 first records in the "customers" table:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "SELECT * FROM mycustomer LIMIT 5"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('Peter', 'Lowstreet 4')
('Amy', 'Apple st 652')
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
```

Start From Another Position

If you want to return five records, starting from the third record, you can use the "OFFSET" keyword:

Start from position 3, and return 5 records:

```
import mysql.connector

mydb = mysql.connector.connect(host = "localhost", user = "root",
passwd = "EHARAmaika@2000", database = "mydatabase")

mycursor = mydb.cursor()

sql = "SELECT * FROM mycustomer LIMIT 5 OFFSET 2"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
('Betty', 'Green Grass 1')
('Richard', 'Sky st 331')
```

join table

Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

TABLE 1 : users [id, name, fav] { id: 1, name: 'John', fav: 154}, { id: 2, name: 'Peter', fav: 154}, { id: 3, name: 'Amy', fav: 155}, { id: 4, name: 'Hannah', fav:}, { id: 5, name: 'Michael', fav:}

TABLE 2 : products [id, name] { id: 154, name: 'Chocolate Heaven' }, { id: 155, name: 'Tasty Lemons' }, { id: 156, name: 'Vanilla Dreams' }

let's create tables for join:

- prepare table 1 : users

1. create table 1

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase')

mycursor = mydb.cursor()

sql = "CREATE TABLE users( ID INT, name varchar(20), fav varchar(5))"

mycursor.execute(sql)
```

1. insert data in table 1

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase')

mycursor = mydb.cursor()

sql = """INSERT INTO users (ID, name, fav) VALUES (%s, %s, %s)"""
val = [
    (1,'John','154'),
    (2,'Peter','154'),
    (3,'Amy','155'),
    (4,'Hannah',None),
    (5,'Micheal',None)
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, " rows inserted.")
```

```
5  rows inserted.
```

1. print table 1

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
```

```
    database = 'mydatabase')

mycursor = mydb.cursor()

sql = "SELECT * FROM users"
mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
    print(x)

(1, 'John', '154')
(2, 'Peter', '154')
(3, 'Amy', '155')
(4, 'Hannah', None)
(5, 'Micheal', None)
```

- prepare table 2 : products

1. create table 2

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase')

mycursor = mydb.cursor()

sql = "CREATE TABLE products(ID INT, name varchar(20))"

mycursor.execute(sql)
```

1. insert data in table 2

```
import mysql.connector

mydb = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    passwd = 'EHARAmaika@2000',
    database = 'mydatabase')

mycursor = mydb.cursor()

sql = "INSERT INTO products VALUES (%s, %s)"
val = [(154,'Chocolate Heaven'),
       (154,'Tasty Lemons'),
       (156, 'Vanilla Dreams')]
```

```
mycursor.executemany(sql,val)

mydb.commit()

print(mycursor.rowcount,"rows inserted.")
```

```
3 rows inserted.
```

1. print table 2

```
import mysql.connector

mydb = mysql.connector.connect(host = 'localhost',
                       user = 'root',
                       passwd = 'EHARAmaika@2000',
                      database = "mydatabase")
mycursor = mydb.cursor()

sql ="SELECT * FROM products"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
(154, 'Chocolate Heaven')
(154, 'Tasty Lemons')
(156, 'Vanilla Dreams')
```

let's see how to use 'JOIN' operations on table 1. users and table 2. products

- These two tables can be combined by using users' fav field and products' id field.

Join users and products to see the name of the users favorite product:

```
import mysql.connector

mydb = mysql.connector.connect(host = 'localhost',
                       user = 'root',
                       passwd = 'EHARAmaika@2000',
                      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)
```

```
('John', 'Tasty Lemons')
('John', 'Chocolate Heaven')
('Peter', 'Tasty Lemons')
('Peter', 'Chocolate Heaven')
```

Note: You can use JOIN instead of INNER JOIN. They will both give you the same result.

LEFT JOIN

In the example above, Hannah, and Michael were excluded from the result, that is because INNER JOIN only shows the records where there is a match. If you want to show all users, even if they do not have a favorite product, use the LEFT JOIN statement:

Select all users and their favorite product:

```
import mysql.connector

mydb = mysql.connector.connect(host = 'localhost',
                       user = 'root',
                       passwd = 'EHARAmaika@2000',
                      database = "mydatabase")
mycursor = mydb.cursor()

sql = "SELECT \
  users.name AS user, \
  products.name AS favorite \
  FROM users \
  LEFT JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('John', 'Tasty Lemons')
('John', 'Chocolate Heaven')
('Peter', 'Tasty Lemons')
('Peter', 'Chocolate Heaven')
```

```
('Amy', None)
('Hannah', None)
('Micheal', None)
```

RIGHT JOIN

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:

Select all products, and the user(s) who have them as their favorite:

```
import mysql.connector

mydb = mysql.connector.connect(host = 'localhost',
                       user = 'root',
                       passwd = 'EHARAmaika@2000',
                      database = "mydatabase")
mycursor = mydb.cursor()

sql = "SELECT \
  users.name AS user, \
  products.name AS favorite \
  FROM users \
  RIGHT JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
    print(x)

('Peter', 'Chocolate Heaven')
('John', 'Chocolate Heaven')
('Peter', 'Tasty Lemons')
('John', 'Tasty Lemons')
(None, 'Vanilla Dreams')
```

you can explore more python techniques for mysql from official python or mysql documantation

Reference

- w3school
- official python documentation