1) Python MySQL Select From

Select from a Table

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

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

```
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)
```

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

Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

Example:

Select only the name and address columns:

```
mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

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

Using the fetchone( ) Method

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:

Example:

Fetch only one row:

```
mycursor.execute(“SELECT * FROM customers”)

myresult=mycursor.fetchone()

print(myresult)
```

2) Python MySQL Where

Select with a Filter

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

Example:
Select record(s) where the address is "Park Lane 38":result:

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

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

3) 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.

Example

Sort the result alphabetically by name: result:

```
sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

Order by DESC

Use the DESC keyword to sort the result in a descending order.

Example:

Sort the result reverse alphabetically by name:

```
sql = "SELECT * FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

4) Python MySQL Delete From By

Delete Record

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

Example:

Delete any record where the address is "Mountain 21"

```
sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

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

Important!: 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 DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

5) Python MySQL Drop Table

Delete a Table

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

Example

Delete the table "customers":

```
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.

Example

Delete the table "customers" if it exists:

```
sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)
```

6) Python MySQL Update Table

Update Table

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

Example

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

```
sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

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

Important!: 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!

7) Python MySQL Limit

Limit the Result

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

Example

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

```
mycursor.execute("SELECT * FROM customers LIMIT 5")

myresult = mycursor.fetchall()

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

Start From Another Position

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

Example

Start from position 3, and return 5 records:

```
mycursor.execute("SELECT * FROM customers LIMIT 5 OFFSET 2")

myresult = mycursor.fetchall()

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

8) Python MySQL Join

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:

users

```
{ 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:}
```

products

```
{ id: 154, name: 'Chocolate Heaven' },
{ id: 155, name: 'Tasty Lemons' },
{ id: 156, name: 'Vanilla Dreams' }
```

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

Example

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

```
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)
```

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:

Example

Select all users and their favorite product:

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

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:

Example:

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

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

Note: Hannah and Michael, who have no favorite product, are not included in the result.