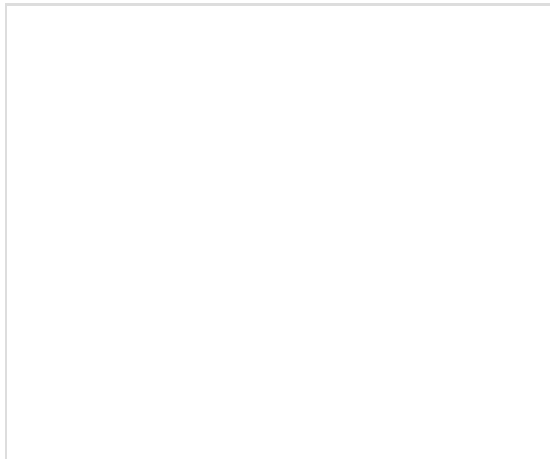


[HOME](#) [START HERE](#) [VIEWS](#) [INDEXES](#) [TRIGGERS](#)[FUNCTIONS](#)[API](#)[AGGREGATE FUNCTIONS](#)[SQLITE PYTHON](#)[DATE FUNCTIONS](#)[SQLITE NODE.JS](#)[STRING FUNCTIONS](#)[SQLITE JAVA](#)[WINDOW FUNCTIONS](#)[SQLITE PHP](#)[TRY IT](#)[Home](#) / [SQLite Python](#) / SQLite Python: Querying Data

SQLite Python: Querying Data

ADVERTISEMENT

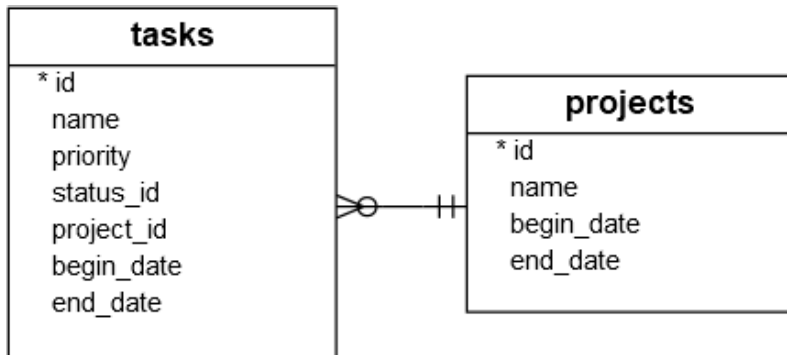


Summary: in this tutorial, we will show you step by step how to query data in SQLite from Python.

To query data in an SQLite database from Python, you use these steps:

1. First, [establish a connection to the SQLite database](#) by creating a Connection object.
2. Next, create a Cursor object using the cursor method of the Connection object.
3. Then, execute a [SELECT](#) statement.
4. After that, call the `fetchall()` method of the cursor object to fetch the data.
5. Finally, loop the cursor and process each row individually.

In the following example, we will use the `tasks` table created in the [creating tables tutorial](#).



First, create a connection to an SQLite database specified by a file:

```

def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
    except Error as e:
        print(e)

    return conn
  
```



This function selects all rows from the tasks table and displays the data:



```
def select_all_tasks(conn):  
    """  
    Query all rows in the tasks table  
    :param conn: the Connection object  
    :return:  
    """  
  
    cur = conn.cursor()  
    cur.execute("SELECT * FROM tasks")  
  
    rows = cur.fetchall()  
  
    for row in rows:  
        print(row)
```

In the `select_all_tasks()` function, we created a cursor, executed the `SELECT` statement, and called the `fetchall()` to fetch all tasks from the `tasks` table.

This function query tasks by priority:



```
def select_task_by_priority(conn, priority):  
    """  
    Query tasks by priority  
    :param conn: the Connection object  
    :param priority:  
    :return:  
    """  
  
    cur = conn.cursor()  
    cur.execute("SELECT * FROM tasks WHERE priority=?", (priority,))  
  
    rows = cur.fetchall()
```

```
for row in rows:
    print(row)
```

In the `select_task_by_priority()` function, we selected the tasks based on a particular priority. The question mark (?) in the query is the placeholder. When the cursor executed the `SELECT` statement, it substituted the question mark (?) by the `priority` argument. The `fetchall()` method fetched all matching tasks by the priority.

This `main()` function creates a connection to the database `C:\sqlite\db\pythonsqlite.db` and calls the functions to query all rows from the `tasks` table and select tasks with priority 1:

```
def main():
    database = r"C:\sqlite\db\pythonsqlite.db"

    # create a database connection
    conn = create_connection(database)
    with conn:

        print("1. Query task by priority:")
        select_task_by_priority(conn, 1)

        print("2. Query all tasks")
        select_all_tasks(conn)
```



Here is the full program:

```
import sqlite3
from sqlite3 import Error

def create_connection(db_file):
    """ create a database connection to the SQLite database
```



```
        specified by the db_file
:param db_file: database file
:return: Connection object or None
"""
conn = None
try:
    conn = sqlite3.connect(db_file)
except Error as e:
    print(e)

return conn
```

```
def select_all_tasks(conn):
    """
    Query all rows in the tasks table
    :param conn: the Connection object
    :return:
    """
    cur = conn.cursor()
    cur.execute("SELECT * FROM tasks")

    rows = cur.fetchall()

    for row in rows:
        print(row)
```

```
def select_task_by_priority(conn, priority):
    """
    Query tasks by priority
    :param conn: the Connection object
```

```
        :param priority:
        :return:
        """

    cur = conn.cursor()
    cur.execute("SELECT * FROM tasks WHERE priority=?", (priority,))

    rows = cur.fetchall()

    for row in rows:
        print(row)

def main():
    database = r"C:\sqlite\db\pythonsqlite.db"


    # create a database connection
    conn = create_connection(database)
    with conn:
        print("1. Query task by priority:")
        select_task_by_priority(conn, 1)

        print("2. Query all tasks")
        select_all_tasks(conn)

if __name__ == '__main__':
    main()
```

In this tutorial, you have learned how to develop a Python program to query data from tables in an SQLite database.

Was this tutorial helpful ?

 Yes

 No

ADVERTISEMENT



Previous

[SQLite Python: Deleting Data](#)

Search this website

GETTING STARTED

[What Is SQLite](#)

[Download & Install SQLite](#)

[SQLite Sample Database](#)

[SQLite Commands](#)

ADVERTISEMENT



SQLITE TUTORIAL

[SQLite Select](#)

[SQLite Order By](#)

[SQLite Select Distinct](#)

[SQLite Where](#)

[SQLite Limit](#)

[SQLite BETWEEN](#)

[SQLite IN](#)

[SQLite Like](#)

[SQLite IS NULL](#)

[SQLite GLOB](#)

[SQLite Join](#)

[SQLite Inner Join](#)

[SQLite Left Join](#)

[SQLite Cross Join](#)

[SQLite Self-Join](#)

[SQLite Full Outer Join](#)

[SQLite Group By](#)

[SQLite Having](#)

[SQLite Union](#)

[SQLite Except](#)

[SQLite Intersect](#)

[SQLite Subquery](#)

[SQLite EXISTS](#)

[SQLite Case](#)

[SQLite Insert](#)

[SQLite Update](#)

[SQLite Delete](#)

[SQLite Replace](#)

[SQLite Transaction](#)

ADVERTISEMENT

SQLITE DATA DEFINITION

[SQLite Data Types](#)

[SQLite Date & Time](#)

[SQLite Create Table](#)

[SQLite Primary Key](#)

[SQLite Foreign Key](#)

[SQLite NOT NULL Constraint](#)

[SQLite UNIQUE Constraint](#)

[SQLite CHECK Constraint](#)

[SQLite AUTOINCREMENT](#)

[SQLite Alter Table](#)

[SQLite Rename Column](#)

[SQLite Drop Table](#)

[SQLite Create View](#)

[SQLite Drop View](#)

[SQLite Index](#)

[SQLite Expression-based Index](#)

[SQLite Trigger](#)

[SQLite VACUUM](#)

[SQLite Transaction](#)

[SQLite Full-text Search](#)

ADVERTISEMENT

SQLITE TOOLS

[SQLite Commands](#)

[SQLite Show Tables](#)

[SQLite Describe Table](#)

[SQLite Dump](#)

[SQLite Import CSV](#)

[SQLite Export CSV](#)

SQLITE FUNCTIONS

[SQLite AVG](#)

[SQLite COUNT](#)

[SQLite MAX](#)

[SQLite MIN](#)

[SQLite SUM](#)

SQLITE INTERFACES

[SQLite PHP](#)

[SQLite Node.js](#)

[SQLite Java](#)

[SQLite Python](#)

ADVERTISEMENT

ABOUT SQLITE TUTORIAL

SQLite Tutorial website helps you master SQLite quickly and easily. It explains the complex concepts in simple and easy-to-understand ways so that you can both understand SQLite fast and know how to apply it in your software development work more effectively.

LOOKING FOR A TUTORIAL...

If you did not find the tutorial that you are looking for, you can use the following search box. In case the tutorial is not available, you can request for it using the [request for a SQLite tutorial form](#).

RECENT TUTORIALS

[SQLite Getting Started](#)

[SQLite Programming Interfaces](#)

[SQLite Concat](#)

[SQLite INSTEAD OF Triggers](#)

[SQLite Join](#)

[SQLite IS NULL](#)

[SQLite Rename Column](#)

[SQLite DROP VIEW](#)

SITE LINKS

[Home](#)

[About](#)

[Contact](#)

[Resources](#)

[Privacy Policy](#)

Copyright © 2020 SQLite Tutorial. All Rights Reserved.