How to Get Rows in a Result Set

The members of the cursor object for getting rows from a result set

After you execute a query, the cursor object contains a result set with the rows returned by the query. Then, to access a row or rows, you can use the fetchone() or fetchall() methods of the cursor object. These methods are summarized below:

Method	Description
`fetchone()`	Returns a tuple containing the next row from the result set. If there is no next row in the result set, this method returns None.
`fetchall()`	Returns a list containing all of the rows in the result set.

How to Get Rows in a Result Set

How to use the fetchone() method to get a row from a table

This example begins by getting a cursor and executing a query that selects the employee in the employees table that has an employeeid of 1. Then, it uses the fetchone() method to get the row that's in the result set and store it in a variable named employee. Because this code is within a with statement for the closing() function, the cursor is closed after the statements are executed.

```
with closing(conn.cursor()) as cursor:
    query = "SELECT * FROM employees WHERE employeeid = ?"
    cursor.execute(query, (1,))
    employee = cursor.fetchone()
```

How to Get Rows in a Result Set

How to use the fetchone() method to get a row from a table

After a row is stored in the employee variable, you can access the columns of the row using an index. For instance, this example prints the name and email for the employee by specifying indexes of 1 and 4 for those columns. This works because name and email are in the columns with the indexes 1 and 4:

```
with closing(conn.cursor()) as cursor:
    query = "SELECT * FROM employees WHERE employeeid = ?"
    cursor.execute(query, (1,))
    employee = cursor.fetchone()

    print("Name: " + employee[1])
    print("Email: " + employee[4])
```

How to Get Rows in a Result Set

How to access columns by name

Because indexes don’t clearly identify the data that’s stored in the columns, this example shows how to access the columns by name. For SQLite, you set the row_factory attribute of the connection object to sqlite3.Row. Them, rather than reference the field by its index, you reference the name of the column. For SQL Server and MySQL, the process is much easier. You simply use dot syntax to access the names of the fields that you want to display.

SQLite

```
conn.row_factory = sqlite3.Row
print("Name: " + employee["name"])
print("Email: " + employee["email"])
```

SQL Server and MySQL

```
print("Name: " + employee.name)
```

How to Get Rows in a Result Set

How to access all of the rows

If you want to access all of the rows in a result, you can use the fetchall() method. Here, the query returns all rows in the employees table. Then, the fetchall() method stores all of these rows in a list named employees. At this point, you can use a for loop to access each employee in the list. In this example, this loop prints all of the employee names and emails:

```
with closing(conn.cursor()) as cursor:
    query = "SELECT * FROM employees"
    cursor.execute(query)
    employees = cursor.fetchall()

    for employee in employees:
        print("Name: " + employee["name"])
        print("Email: " + employee["email"])
        print("")
```