

# Modulo 6

## Contents

1	Criando meu primeiro arquivo do SQLite (db.sqlite3)	1
2	Criando minha primeira tabela no SQLite3 (DBeaver opcional)	1
3	inserindo valores (INSERT INTO), DELETE sem WHERE e zerando a sqlite	1
4	Usando placeholders	2
5	Inserindo vários valores com execute many	2
6	execute e executemany com dicionários e lista de dicionários	2
7	SELECT do SQL com fetch no SQLite3 do Python	3
8	O que é CRUD + DELETE com e sem WHERE no SQLite3 do Python	3
9	DELETE no SQLite do Python	4
10	UPDATE no SQLite com Python	4

## 1 Criando meu primeiro arquivo do SQLite (db.sqlite3)

```
import sqlite3
from pathlib import Path

ROOT_DIR = Path(__file__).parent
DB_NAME = 'db.sqlite3'
DB_FILE = ROOT_DIR / DB_NAME

connection = sqlite3.connect(DB_FILE)
cursor = connection.cursor()

# SQL

cursor.close()
connection.close()
```

## 2 Criando minha primeira tabela no SQLite3 (DBeaver opcional)

```
ROOT_DIR = Path(__file__).parent
DB_NAME = 'db.sqlite3'
DB_FILE = ROOT_DIR / DB_NAME
TABLE_NAME = 'customers'

connection = sqlite3.connect(DB_FILE)
cursor = connection.cursor()

# SQL
cursor.execute(
    f'CREATE TABLE IF NOT EXISTS {TABLE_NAME}',
    (,
    'id INTEGER PRIMARY KEY AUTOINCREMENT,'
    'name TEXT,'
    'weight REAL'
    ),
)
connection.commit()

cursor.close()
connection.close()
```

## 3 inserindo valores (INSERT INTO), DELETE sem WHERE e zerando a sqlite

```
connection = sqlite3.connect(DB_FILE)
cursor = connection.cursor()

# CUIDADO: fazendo delete sem where
cursor.execute(
```

```

        f'DELETE FROM {TABLE_NAME}'
    )
    cursor.execute(
        f'DELETE FROM sqlite_sequence WHERE name="{TABLE_NAME}"'
    )
    connection.commit()

# Cria a tabela
cursor.execute(
    f'CREATE TABLE IF NOT EXISTS {TABLE_NAME}'
    ,(,
)
connection.commit()

# Registrar valores nas colunas da tabela
# CUIDADO: sql injection
cursor.execute(
    f'INSERT INTO {TABLE_NAME} '
    '(id, name, weight) '
    'VALUES '
    '(NULL, "Helena", 4), (NULL, "Eduardo", 10)'
)
connection.commit()

cursor.close()
connection.close()

```

## 4 Usando placeholders

Esse código é uma continuação do código anterior

```

        connection.commit()

# Registrar valores nas colunas da tabela
# CUIDADO: sql injection
cursor.execute(
    sql = (
        f'INSERT INTO {TABLE_NAME} '
        '(id, name, weight) '
        '(name, weight) '
        'VALUES '
        '(NULL, "Helena", 4), (NULL, "Eduardo", 10)'
        '(?, ?)'
    )
)
cursor.execute(sql, ['Joana', 4])
connection.commit()
print(sql)

cursor.close()
connection.close()

```

## 5 Inserindo vários valores com execute many

```

        'VALUES '
        '(?, ?)'
    )
    cursor.execute(sql, ['Joana', 4])
    # cursor.execute(sql, ['Joana', 4])
    cursor.executemany(
        sql,
        (
            ('Joana', 4), ('Luiz', 5)
        )
    )
    connection.commit()
    print(sql)

```

## 6 execute e executemany com dicionários e lista de dicionários

```

        f'INSERT INTO {TABLE_NAME} '
        '(name, weight) '
        'VALUES '
        '(?, ?)'
        '(:nome, :peso)'
    )
    # cursor.execute(sql, ['Joana', 4])
    cursor.executemany(
        sql,

```

```

        (
            ('Joana', 4), ('Luiz', 5)
        )
    )
)
# cursor.executemany(
#     sql,
#     (
#         ('Joana', 4), ('Luiz', 5)
#     )
# )
cursor.execute(sql, {'nome': 'Sem nome', 'peso': 3})
cursor.executemany(sql, (
    {'nome': 'Joãozinho', 'peso': 3},
    {'nome': 'Maria', 'peso': 2},
    {'nome': 'Helena', 'peso': 4},
    {'nome': 'Joana', 'peso': 5},
))
connection.commit()
print(sql)

```

## 7 SELECT do SQL com fetch no SQLite3 do Python

main.py

```

        {'nome': 'Joana', 'peso': 5},
    ))
    connection.commit()
    print(sql)

    cursor.close()
    connection.close()

    if __name__ == '__main__':
        print(sql)

```

select.py

```

import sqlite3

from main import DB_FILE, TABLE_NAME

connection = sqlite3.connect(DB_FILE)
cursor = connection.cursor()

cursor.execute(
    f'SELECT * FROM {TABLE_NAME}'
)

for row in cursor.fetchall():
    _id, name, weight = row
    print(_id, name, weight)

print()

cursor.execute(
    f'SELECT * FROM {TABLE_NAME} '
    'WHERE id = "3"'
)
row = cursor.fetchone()
_id, name, weight = row
print(_id, name, weight)

cursor.close()
connection.close()

```

## 8 O que é CRUD + DELETE com e sem WHERE no SQLite3 do Python

```

        connection = sqlite3.connect(DB_FILE)
    cursor = connection.cursor()

    # CRUD - Create Read Update Delete
    # SQL - INSERT SELECT UPDATE DELETE

    # CUIDADO: fazendo delete sem where
    cursor.execute(
        f'DELETE FROM {TABLE_NAME}'
    )

    # DELETE mais cuidadoso
    cursor.execute(

```

```

        f'DELETE FROM sqlite_sequence WHERE name="{TABLE_NAME}"',
    )
    @@ -51,7 +56,6 @@
        {'nome': 'Joana', 'peso': 5},
    ))
    connection.commit()

    cursor.close()
    connection.close()

```

## 9 DELETE no SQLite do Python

```

        {'nome': 'Joana', 'peso': 5},
    ))
    connection.commit()
    cursor.close()
    connection.close()

if __name__ == '__main__':
    print(sql)

    cursor.execute(
        f'DELETE FROM {TABLE_NAME} '
        'WHERE id = "3"'
    )
    cursor.execute(
        f'DELETE FROM {TABLE_NAME} '
        'WHERE id = 1'
    )
    connection.commit()

    cursor.execute(
        f'SELECT * FROM {TABLE_NAME}'
    )

    for row in cursor.fetchall():
        _id, name, weight = row
        print(_id, name, weight)

    cursor.close()
    connection.close()

```

## 10 UPDATE no SQLite com Python

```

    )
    connection.commit()

    cursor.execute(
        f'UPDATE {TABLE_NAME} '
        'SET name="QUALQUER", weight=67.89 '
        'WHERE id = 2'
    )
    connection.commit()

    cursor.execute(
        f'SELECT * FROM {TABLE_NAME}'
    )

```