Modulo 6

Contents

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 codigo é uma continuação do codigo 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}'
        )
```

11 Pymysql

main.py

```
        import pymysql
        import dotenv #type:ignore
        import os
        dotenv.load_dotenv()

        connection = pymysql.connect(
            host =os.environ['MYSQL_HOST'],
            user = os.environ['MYSQL_USER'],
            password =os.environ['MYSQL_PASSWORD'],
            database =os.environ['MYSQL_DATABASE']
        )

        print(os.environ['MYSQL_HOST'])
        with connection:
        with connection.cursor() as cursor:

    cursor.close()
```

docker-compose.yml

```
        version: '3.9'
    services:
    mysql_206:
        env_file:
        - .env
        container_name: mysql_206
        hostname: mysql_206
        image: mysql:8
        restart: always
        command:
        - --authentication-policy=mysql_native_password
        - --character-set-server=utf8mb4
        - --collation-server=utf8mb4_unicode_ci
        - --innodb_force_recovery=0
        volumes:
        - ./mysql_206:/var/lib/mysql
        ports:
        - 3306:3306
        environment:

        TZ: America/Sao_Paulo
```

.env-example

```
        MYSQL_ROOT_PASSWORD = 'CHANGE-ME'
        MYSQL_DATABASE = 'CHANGE-ME'
        MYSQL_USER = 'CHANGE-ME'
        MYSQL_PASSWORD = 'CHANGE-ME'
        MYSQL_HOST = 'CHANGE-ME'
```

12 CREATE TABLE para criar tabela com PRIMARY KEY no PyMySQL

```
    with connection:
with connection.cursor() as cursor:
    # SQL
    cursor.execute(  # type: ignore
        'CREATE TABLE IF NOT EXISTS customers ('
        'id INT NOT NULL AUTO_INCREMENT, '
        'nome VARCHAR(50) NOT NULL, '
        'idade INT NOT NULL, '
        'PRIMARY KEY (id)'
        ') '
    )
    print(cursor)
```

13 TRUNCATE e INSERT p/ limpar e criar valores na tabela com um ou mais colunas

```
    import dotenv
import pymysql

TABLE_NAME = 'customers'
```

```
dotenv.load_dotenv()

connection = pymysql.connect(
    host=os.environ['MYSQL_HOST'],
    user=os.environ['MYSQL_USER'],
    password=os.environ['MYSQL_PASSWORD'],
    database=os.environ['MYSQL_DATABASE'],
    charset='utf8mb4'
)

with connection:
    with connection.cursor() as cursor:
        cursor.execute(  # type: ignore
            'CREATE TABLE IF NOT EXISTS customers ('
            f'CREATE TABLE IF NOT EXISTS {TABLE_NAME} ('
            'id INT NOT NULL AUTO_INCREMENT, '
            'nome VARCHAR(50) NOT NULL, '
            'idade INT NOT NULL, '
            'PRIMARY KEY (id)'
            ') '
        )
        print(cursor)
        # CUIDADO: ISSO LIMPA A TABELA
        cursor.execute(f'TRUNCATE TABLE {TABLE_NAME}')  # type: ignore
    connection.commit()

    # Começo a manipular dados a partir daqui

    with connection.cursor() as cursor:
        cursor.execute(  # type: ignore
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
        )
        cursor.execute(  # type: ignore
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
        )
        result = cursor.execute(  # type: ignore
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
        )
        print(result)
    connection.commit()
```

14 Evite SQL Injection ao usar placeholders para enviar valores

```
        # Começo a manipular dados a partir daqui

    with connection.cursor() as cursor:
        cursor.execute(  # type: ignore
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
        )
        cursor.execute(  # type: ignore
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
        )
        result = cursor.execute(  # type: ignore
        sql = (
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) VALUES ("Luiz", 25) '
            '(nome, idade) '
            'VALUES '
            '(%s, %s) '
        )
        data = ('Luiz', 18)
        result = cursor.execute(sql, data)  # type: ignore
        print(sql, data)
        print(result)
    connection.commit()
```

15 Inserindo valores usando dicionários ao invés de iteráveis

```
        )
        data = ('Luiz', 18)
        result = cursor.execute(sql, data)  # type: ignore
        print(sql, data)
        # print(sql, data)
```

```
        # print(result)
    connection.commit()

    with connection.cursor() as cursor:
        sql = (
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) '
            'VALUES '
            '(%(name)s, %(age)s) '
        )
        data2 = {
            "age": 37,
            "name": "Le",
        }
        result = cursor.execute(sql, data2)  # type: ignore
        print(sql)
        print(data2)
        print(result)
    connection.commit()
```

16 Lendo valores com SELECT, cursor.execute e cursor.fetchall no PyMySQL

```
        # PyMySQL - um cliente MySQL feito em Python Puro
# Doc: https://pymysql.readthedocs.io/en/latest/
# Pypy: https://pypi.org/project/pymysql/
# GitHub: https://github.com/PyMySQL/PyMySQL
import os

import dotenv
import pymysql

TABLE_NAME = 'customers'

dotenv.load_dotenv()

connection = pymysql.connect(
    host=os.environ['MYSQL_HOST'],
    user=os.environ['MYSQL_USER'],
    password=os.environ['MYSQL_PASSWORD'],
    database=os.environ['MYSQL_DATABASE'],
    charset='utf8mb4'
)

with connection:
    with connection.cursor() as cursor:
        cursor.execute(  # type: ignore
            f'CREATE TABLE IF NOT EXISTS {TABLE_NAME} ('
            'id INT NOT NULL AUTO_INCREMENT, '
            'nome VARCHAR(50) NOT NULL, '
            'idade INT NOT NULL, '
            'PRIMARY KEY (id)'
            ') '
        )
        # CUIDADO: ISSO LIMPA A TABELA
        cursor.execute(f'TRUNCATE TABLE {TABLE_NAME}')  # type: ignore
    connection.commit()

    # Começo a manipular dados a partir daqui

    # Inserindo um valor usando placeholder e um iterável
    with connection.cursor() as cursor:
        sql = (
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) '
            'VALUES '
            '(%s, %s) '
        )
        data = ('Luiz', 18)
        result = cursor.execute(sql, data)  # type: ignore
        # print(sql, data)
        # print(result)
    connection.commit()

    # Inserindo um valor usando placeholder e um dicionário
    with connection.cursor() as cursor:
        sql = (
            f'INSERT INTO {TABLE_NAME} '
            '(nome, idade) '
            'VALUES '
            '(%(name)s, %(age)s) '
```

```
    )
    data2 = {
        "age": 37,
        "name": "Le",
    }
    result = cursor.execute(sql, data2)  # type: ignore
    # print(sql)
    # print(data2)
    # print(result)
connection.commit()

# Inserindo vários valores usando placeholder e um tupla de dicionários
with connection.cursor() as cursor:
    sql = (
        f'INSERT INTO {TABLE_NAME} '
        '(nome, idade) '
        'VALUES '
        '(%(name)s, %(age)s) '
    )
    data3 = (
        {"name": "Sah", "age": 33, },
        {"name": "Julia", "age": 74, },
        {"name": "Rose", "age": 53, },
    )
    result = cursor.executemany(sql, data3)  # type: ignore
    # print(sql)
    # print(data3)
    # print(result)
connection.commit()

# Inserindo vários valores usando placeholder e um tupla de tuplas
with connection.cursor() as cursor:
    sql = (
        f'INSERT INTO {TABLE_NAME} '
        '(nome, idade) '
        'VALUES '
        '(%s, %s) '
    )
    data4 = (
        ("Siri", 22, ),
        ("Helena", 15, ),
    )
    result = cursor.executemany(sql, data4)  # type: ignore
    # print(sql)
    # print(data4)
    # print(result)
connection.commit()

# Lendo os valores com SELECT
with connection.cursor() as cursor:
    sql = (
        f'SELECT * FROM {TABLE_NAME} '
    )
    cursor.execute(sql)  # type: ignore
    data5 = cursor.fetchall()  # type: ignore

    for row in data5:
        print(row)
```

17 Lendo valores com WHERE (mais uma vez, explico cuidados com SQL)

```
    data4 = (
        ("Siri", 22, ),
        ("Helena", 15, ),
        ("Luiz", 18, ),
    )
    result = cursor.executemany(sql, data4)  # type: ignore
    # print(sql)

# Lendo os valores com SELECT
with connection.cursor() as cursor:
    menor_id = int(input('Digite o menor id: '))
    maior_id = int(input('Digite o maior id: '))

    sql = (
        f'SELECT * FROM {TABLE_NAME} '
        'WHERE id BETWEEN %s AND %s  '
    )
    cursor.execute(sql)  # type: ignore
```

```
        cursor.execute(sql, (menor_id, maior_id))  # type: ignore
        print(cursor.mogrify(sql, (menor_id, maior_id)))  # type: ignore
        data5 = cursor.fetchall()  # type: ignore

        for row in data5:
```

18 Apagando valores com DELETE, WHERE e placeholders no PyMySQL

```
        # Lendo os valores com SELECT
        with connection.cursor() as cursor:
            menor_id = int(input('Digite o menor id: '))
            maior_id = int(input('Digite o maior id: '))
            # menor_id = int(input('Digite o menor id: '))
            # maior_id = int(input('Digite o maior id: '))
            menor_id = 2
            maior_id = 4

            sql = (
                f'SELECT * FROM {TABLE_NAME} '
                'WHERE id BETWEEN %s AND %s  '
            )

            cursor.execute(sql, (menor_id, maior_id))  # type: ignore
            print(cursor.mogrify(sql, (menor_id, maior_id)))  # type: ignore
            # print(cursor.mogrify(sql, (menor_id, maior_id)))  # type: ignore
            data5 = cursor.fetchall()  # type: ignore

            for row in data5:
            # for row in data5:
            # print(row)

        # Apagando com DELETE, WHERE e placeholders no PyMySQL
        with connection.cursor() as cursor:
            sql = (
                f'DELETE FROM {TABLE_NAME} '
                'WHERE id = %s'
            )
            print(cursor.execute(sql, (1,)))  # type: ignore
            connection.commit()

            cursor.execute(f'SELECT * FROM {TABLE_NAME} ')  # type: ignore

            for row in cursor.fetchall():  # type: ignore
                print(row)
```

19 Editando com UPDATE, WHERE e placeholders no PyMySQL

```
            f'DELETE FROM {TABLE_NAME} '
            'WHERE id = %s'
        )
        print(cursor.execute(sql, (1,)))  # type: ignore
        cursor.execute(sql, (1,))  # type: ignore
        connection.commit()

        cursor.execute(f'SELECT * FROM {TABLE_NAME} ')  # type: ignore

        # for row in cursor.fetchall():  # type: ignore
        #     print(row)

# Editando com UPDATE, WHERE e placeholders no PyMySQL
with connection.cursor() as cursor:
    sql = (
        f'UPDATE {TABLE_NAME} '
        'SET nome=%s, idade=%s '
        'WHERE id=%s'
    )
    cursor.execute(sql, ('Eleonor', 102, 4))  # type: ignore
    cursor.execute(f'SELECT * FROM {TABLE_NAME} ')  # type: ignore

    for row in cursor.fetchall():  # type: ignore
        print(row)
connection.commit()
```

20 Trocando o cursor para retornar dicionários - pymysql.cursors.DictCursor

```
import dotenv
```

```
import pymysql
import pymysql.cursors

TABLE_NAME = 'customers'

@@ -16,7 +17,8 @@
    user=os.environ['MYSQL_USER'],
    password=os.environ['MYSQL_PASSWORD'],
    database=os.environ['MYSQL_DATABASE'],
    charset='utf8mb4'
    charset='utf8mb4',
    cursorclass=pymysql.cursors.DictCursor,
)

with connection:
@@ -148,6 +150,10 @@
        cursor.execute(sql, ('Eleonor', 102, 4))  # type: ignore
        cursor.execute(f'SELECT * FROM {TABLE_NAME} ')  # type: ignore

        # for row in cursor.fetchall():  # type: ignore
        #     _id, name, age = row
        #     print(_id, name, age)

        for row in cursor.fetchall():  # type: ignore
            print(row)
    connection.commit()
```

21 rowcount, rownumber e lastrowid para detalhes de consultas executadas

```
        import pymysql.cursors

TABLE_NAME = 'customers'
CURRENT_CURSOR = pymysql.cursors.SSDictCursor
CURRENT_CURSOR = pymysql.cursors.DictCursor

dotenv.load_dotenv()

@@ -152,18 +152,26 @@
            'WHERE id=%s'
        )
        cursor.execute(sql, ('Eleonor', 102, 4))
        cursor.execute(f'SELECT * FROM {TABLE_NAME} ')

        print('For 1: ')
        for row in cursor.fetchall_unbuffered():
            print(row)
        cursor.execute(
            f'SELECT id from {TABLE_NAME} ORDER BY id DESC LIMIT 1'
        )
        lastIdFromSelect = cursor.fetchone()

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

            if row['id'] >= 5:
                break
        data6 = cursor.fetchall()

        print()
        print('For 2: ')
        # cursor.scroll(-1)
        for row in cursor.fetchall_unbuffered():
        for row in data6:
            print(row)

        print('resultFromSelect', resultFromSelect)
        print('len(data6)', len(data6))
        print('rowcount', cursor.rowcount)
        print('lastrowid', cursor.lastrowid)
        print('lastrowid na mão', lastIdFromSelect)

        cursor.scroll(0, 'absolute')
        print('rownumber', cursor.rownumber)

    connection.commit()
```

22 SSCursor, SSDictCursor e scroll para conjuntos de dados muito grandes

```
        # Pypy: https://pypi.org/project/pymysql/
# GitHub: https://github.com/PyMySQL/PyMySQL
```

```
import os
from typing import cast

import dotenv
import pymysql
import pymysql.cursors

TABLE_NAME = 'customers'
CURRENT_CURSOR = pymysql.cursors.SSDictCursor

dotenv.load_dotenv()

@@ -18,7 +20,7 @@
    password=os.environ['MYSQL_PASSWORD'],
    database=os.environ['MYSQL_DATABASE'],
    charset='utf8mb4',
    cursorclass=pymysql.cursors.DictCursor,
    cursorclass=CURRENT_CURSOR,
)

with connection:
@@ -142,18 +144,26 @@

    # Editando com UPDATE, WHERE e placeholders no PyMySQL
    with connection.cursor() as cursor:
        cursor = cast(CURRENT_CURSOR, cursor)

        sql = (
            f'UPDATE {TABLE_NAME} '
            'SET nome=%s, idade=%s '
            'WHERE id=%s'
        )
        cursor.execute(sql, ('Eleonor', 102, 4))  # type: ignore
        cursor.execute(f'SELECT * FROM {TABLE_NAME} ')  # type: ignore
        cursor.execute(sql, ('Eleonor', 102, 4))
        cursor.execute(f'SELECT * FROM {TABLE_NAME} ')

        # for row in cursor.fetchall():  # type: ignore
        #     _id, name, age = row
        #     print(_id, name, age)
        print('For 1: ')
        for row in cursor.fetchall_unbuffered():
            print(row)

            if row['id'] >= 5:
                break

        for row in cursor.fetchall():  # type: ignore
        print()
        print('For 2: ')
        # cursor.scroll(-1)
        for row in cursor.fetchall_unbuffered():
            print(row)
    connection.commit()
```