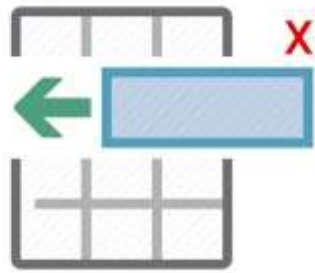


# EMPREGAR AS FUNCIONALIDADES PARA RECUPERAÇÃO DE REGISTROS EM TABELAS

## SELEÇÃO DE REGISTROS DE UMA TABELA

Neste módulo, aprenderemos a recuperar os registros presentes no banco de dados. Partiremos de consultas mais simples, utilizando apenas uma tabela, até consultas mais sofisticadas, envolvendo os relacionamentos entre tabelas.

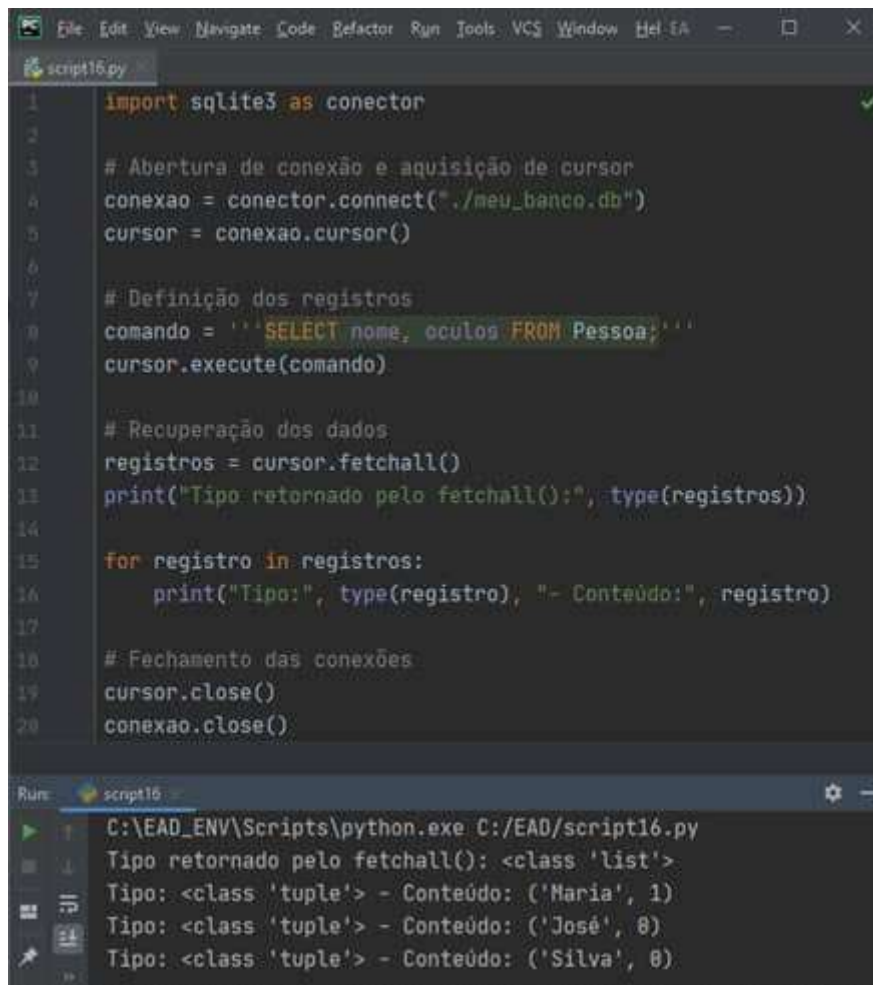


Para seleccionar e recuperar registros de um banco de dados, utilizamos o comando SQL **SELECT**. Sua sintaxe é a seguinte:

```
SELECT      coluna1,      coluna2,      ...      FROM      tabela1  
WHERE [condição];
```

Assim como nos outros comandos, podemos utilizar uma string sem delimitadores, uma string com o delimitador “?” ou uma string com argumentos nomeados para a condição da cláusula **WHERE**.

No exemplo a seguir, Figura 20, vamos mostrar como recuperar todos os registros da tabela **Pessoa**.



```
1 import sqlite3 as conector
2
3 # Abertura de conexão e aquisição de cursor
4 conexao = conector.connect("./meu_banco.db")
5 cursor = conexao.cursor()
6
7 # Definição dos registros
8 comando = 'SELECT nome, olhos FROM Pessoa;'
9 cursor.execute(comando)
10
11 # Recuperação dos dados
12 registros = cursor.fetchall()
13 print("Tipo retornado pelo fetchall():", type(registros))
14
15 for registro in registros:
16     print("Tipo:", type(registro), "- Conteúdo:", registro)
17
18 # Fechamento das conexões
19 cursor.close()
20 conexao.close()
```

Run: script16

C:\EAD\_ENV\Scripts\python.exe C:/EAD/script16.py

Tipo retornado pelo fetchall(): <class 'list'>

Tipo: <class 'tuple'> - Conteúdo: ('Maria', 1)

Tipo: <class 'tuple'> - Conteúdo: ('José', 8)

Tipo: <class 'tuple'> - Conteúdo: ('Silva', 8)

Figura: 20.

Após criar uma conexão e obter um cursor, criamos o comando SQL “SELECT” na linha 8 e destacado a seguir:

***SELECT nome, olhos FROM Pessoa;***

Observe que estamos selecionando todas as pessoas da tabela, visto que não há clausulas **WHERE**. Porém, estamos recuperando apenas os dados das colunas nome e olhos.

Executamos o comando na linha 9 e utilizamos o método fetchall do cursor para recuperar os registros selecionados.

Atribuímos o retorno do método à variável registros.

O objeto retornado pelo método fetchall é do tipo lista, impresso pela linha 13 e que pode ser observado pelo console.

Na linha 15, iteramos sobre os elementos retornados e, na linha 16, imprimimos o tipo e o conteúdo dos registros.

Observe que cada registro é uma tupla, composta pelos atributos nome e olhos da entidade Pessoa.

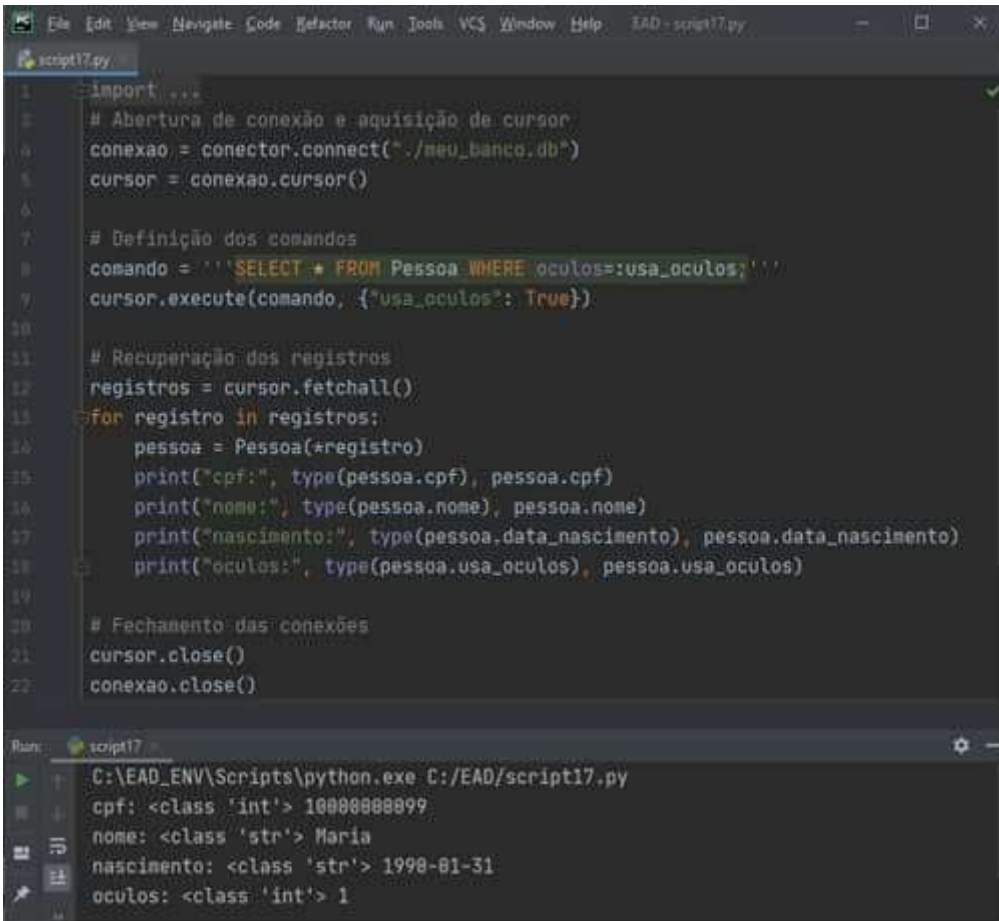
Os registros são sempre retornados em forma de tupla, mesmo que contenham apenas um atributo!

Ao final, fechamos o cursor e a conexão.

## Atenção

Como o SQLite não cria uma transação para o comando SELECT, não é necessário executar o commit.

No exemplo a seguir, Figura 21, vamos criar uma consulta para retornar as pessoas que usam óculos. Observe como ficou o exemplo.

The image shows a screenshot of a code editor window titled 'script17.py'. The code is in Python and uses the sqlite3 module. It connects to a database file named 'meu\_banco.db', executes a SQL query to select all records from a table named 'Pessoa' where 'usa\_olhos' is 1, fetches all records, and prints the details of each record. The output window at the bottom shows the execution results for a single record: cpf: 10000000099, nome: Maria, nascimento: 1990-01-31, and olhos: 1.

```
1 import ...
2 # Abertura de conexão e aquisição de cursor
3 conexao = conector.connect("../meu_banco.db")
4 cursor = conexao.cursor()
5
6
7 # Definição dos comandos
8 comando = '''SELECT * FROM Pessoa WHERE olhos=:usa_olhos;'''
9 cursor.execute(comando, {"usa_olhos": True})
10
11 # Recuperação dos registros
12 registros = cursor.fetchall()
13 for registro in registros:
14     pessoa = Pessoa(*registro)
15     print("cpf:", type(pessoa.cpf), pessoa.cpf)
16     print("nome:", type(pessoa.nome), pessoa.nome)
17     print("nascimento:", type(pessoa.data_nascimento), pessoa.data_nascimento)
18     print("olhos:", type(pessoa.usa_olhos), pessoa.usa_olhos)
19
20 # Fechamento das conexões
21 cursor.close()
22 conexao.close()
```

Run: script17

C:\EAD\_ENV\Scripts\python.exe C:/EAD/script17.py

cpf: <class 'int'> 10000000099

nome: <class 'str'> Maria

nascimento: <class 'str'> 1990-01-31

olhos: <class 'int'> 1

Figura: 21.

Após abrir a conexão e criar um cursor, definimos o comando para selecionar apenas as pessoas que usam óculos.

Para isso, definimos o comando SQL da linha 8, que, após executado, fica da seguinte forma:

```
SELECT * FROM Pessoa WHERE olhos=1;
```

Observe que utilizamos o asterisco (\*) para representar quais dados desejamos receber. No SQL, o asterisco representa todas as colunas.

Na linha 12, recuperamos todos os dados selecionados utilizando o fetchall, que foram iterados na linha 13.

Para cada registro retornado, que é uma tupla com os atributos cpf, nome, nascimento e olhos, nessa ordem, criamos um objeto do tipo Pessoa, na linha 14.

Observe que utilizamos o operador \* do Python. Esse operador “desempacota” um iterável, passando cada elemento como um argumento para uma função ou construtor.

Como sabemos que a ordem das colunas é a mesma ordem dos parâmetros do construtor da classe Pessoa, garantimos que vai funcionar corretamente.

Das linhas 15 a 18, imprimimos cada atributo do registro e seu respectivo tipo.

Ao final do script fechamos a conexão e o cursor.

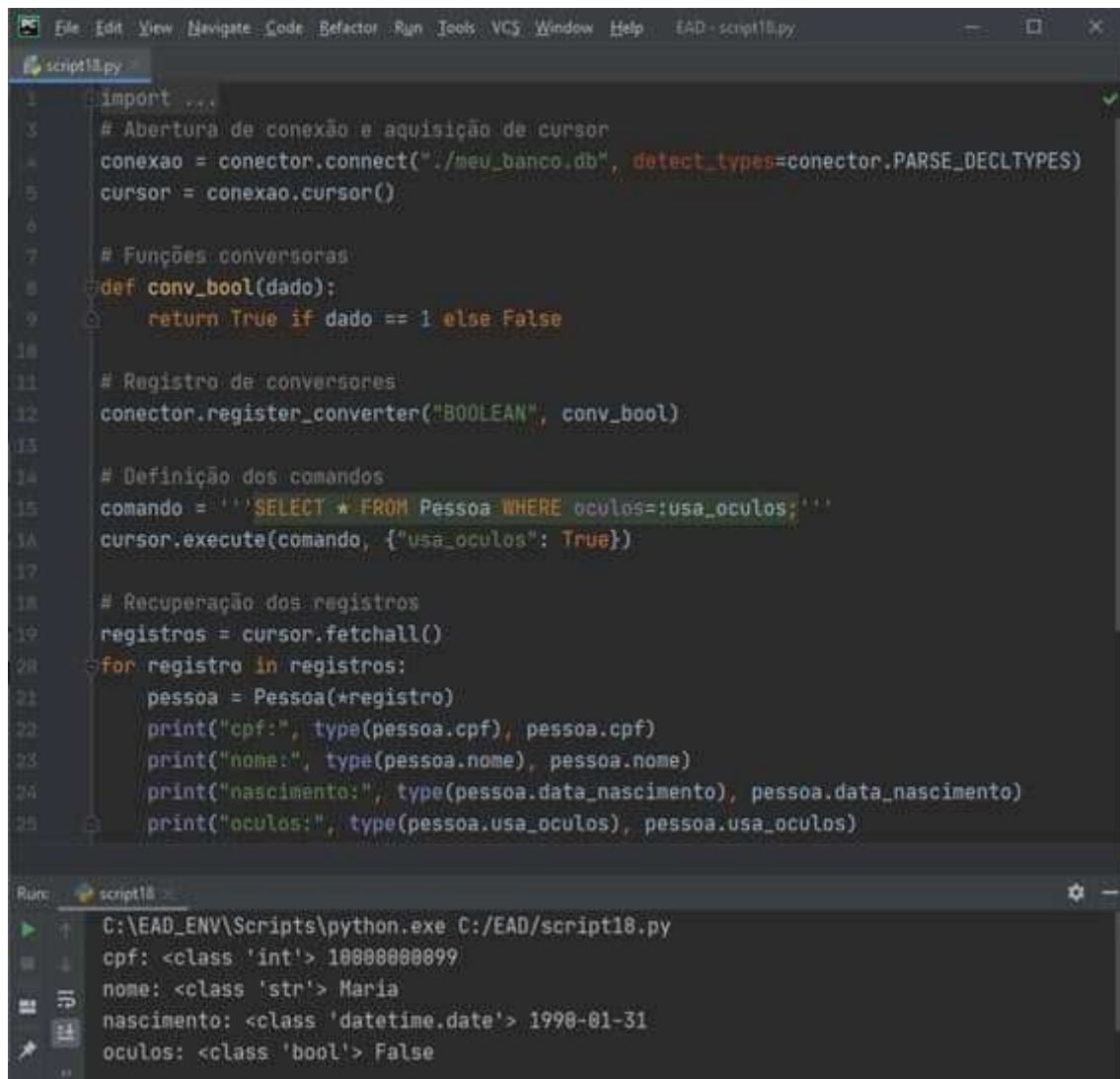
Verifique que os dados dos atributos nascimento e olhos estão exatamente como no banco de dados, o nascimento é uma string e o olhos é do tipo inteiro.

## **Comentário**

Se estivéssemos utilizando o banco de dados **PostgreSQL** com o conector psycopg2, como os tipos **DATE** e **BOOLEAN** são suportados, esses valores seriam convertidos para o tipo correto.

O conector sqlite3 também nos permite fazer essa conversão automaticamente, mas precisamos fazer algumas configurações a mais.

No exemplo a seguir, Figura 22, vamos mostrar como fazer a conversão de datas e booleanos.

The image shows a screenshot of a Python script editor window titled 'script18.py'. The script is written in Python and uses the sqlite3 module to connect to a database. It defines a function 'conv\_bool' to convert database boolean values to Python boolean values. It registers this function with the sqlite3 connector. The script then executes a SQL query to select all records from a table named 'Pessoa' where 'usa\_olhos' is not null. It fetches all records and prints the details of each record, including CPF, name, birth date, and whether they use glasses. The output of the script is shown in the Run console at the bottom, displaying the data for a person named Maria born on 1998-01-31, who does not use glasses.

```
1 import ...
3 # Abertura de conexão e aquisição de cursor
4 conexao = conector.connect("../meu_banco.db", detect_types=conector.PARSE_DECLTYPES)
5 cursor = conexao.cursor()
6
7 # Funções conversoras
8 def conv_bool(dado):
9     return True if dado == 1 else False
10
11 # Registro de conversores
12 conector.register_converter("BOOLEAN", conv_bool)
13
14 # Definição dos comandos
15 comando = 'SELECT * FROM Pessoa WHERE olhos=:usa_olhos;'
16 cursor.execute(comando, {"usa_olhos": True})
17
18 # Recuperação dos registros
19 registros = cursor.fetchall()
20 for registro in registros:
21     pessoa = Pessoa(*registro)
22     print("cpf:", type(pessoa.cpf), pessoa.cpf)
23     print("nome:", type(pessoa.nome), pessoa.nome)
24     print("nascimento:", type(pessoa.data_nascimento), pessoa.data_nascimento)
25     print("olhos:", type(pessoa.usa_olhos), pessoa.usa_olhos)
```

Run: script18

C:\EAD\_ENV\Scripts\python.exe C:/EAD/script18.py  
cpf: <class 'int'> 100000000099  
nome: <class 'str'> Maria  
nascimento: <class 'datetime.date'> 1998-01-31  
olhos: <class 'bool'> False

Figura: 22.

A primeira modificação ocorre nos parâmetros da criação da conexão. Precisamos passar o argumento **PARSE\_DECLTYPES** para o parâmetro **detect\_types** da função **connect**. Observe como ficou a linha 4.

Isso indica que o conector deve tentar fazer uma conversão dos dados, tomando como base o tipo da coluna declarada no **CREATE TABLE**.

## Comentário

Os tipos **DATE** e **TIMESTAMP** já possuem conversores embutidos no **sqlite3**, porém, o tipo **BOOLEAN** não.

Para informar ao conector como fazer a conversão do tipo **BOOLEAN**, precisamos definir e registrar a função conversora utilizando a função interna **register\_converter** do **sqlite3**.

A função ***register\_converter*** espera, como primeiro parâmetro, uma string com o tipo da coluna a ser convertido e, como segundo parâmetro, uma função que recebe o dado e retorna esse dado convertido.

Na linha 12, chamamos a função `register_converter`, passando a string “BOOLEAN” e a função `conv_bool` (converter booleano) como argumentos.

A função `conv_bool`, definida na linha 9, retorna `True` para o caso do dado ser 1, ou `False`, caso contrário. Com isso, convertemos os inteiros 0 e 1 para os booleanos `True` e `False`.

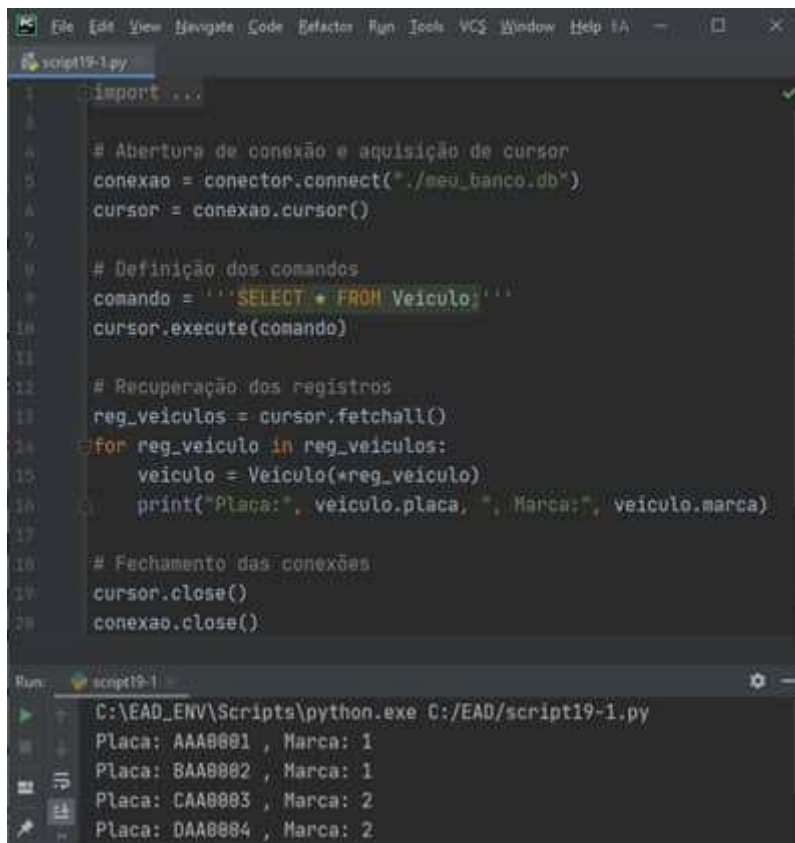
O restante do script é igual ao anterior, porém, os dados estão convertidos para o tipo correto.

Verifique a saída do console e observe os tipos dos atributos nascimento e olhos. Agora são das classes `date` e `bool`!

## SELEÇÃO DE REGISTROS UTILIZANDO JUNÇÃO

Na seção anterior, aprendemos a selecionar registros de apenas uma tabela, mas como podemos buscar registros de tabelas relacionadas?

No exemplo a seguir, Figura 23, vamos buscar os veículos e suas respectivas marcas. Vamos fazer isso por meio da junção de tabelas.



```
1 import ...
2
3 # Abertura de conexão e aquisição de cursor
4 conexao = conector.connect("./meu_banco.db")
5 cursor = conexao.cursor()
6
7 # Definição dos comandos
8 comando = 'SELECT * FROM Veiculo;'
9 cursor.execute(comando)
10
11 # Recuperação dos registros
12 reg_veiculos = cursor.fetchall()
13
14 for reg_veiculo in reg_veiculos:
15     veiculo = Veiculo(*reg_veiculo)
16     print("Placa:", veiculo.placa, ", Marca:", veiculo.marca)
17
18 # Fechamento das conexões
19 cursor.close()
20 conexao.close()
```

Run: script19-1

```
C:\EAD_ENV\Scripts\python.exe C:/EAD/script19-1.py
Placa: AAA8881 , Marca: 1
Placa: BAA8882 , Marca: 1
Placa: CAA8883 , Marca: 2
Placa: DAA8884 , Marca: 2
```

Figura: 23.

Após abrir a conexão e obter um cursor, selecionamos todos os registros da tabela Veiculo, utilizando o comando da linha 9, executado na linha 10.

Na linha 13, recuperamos todos os registros de veículos utilizando o método fetchall, que foram iterados na linha 14.

Na linha 15, criamos um objeto do tipo Veiculo utilizando o operador \* e imprimimos os atributos placa e marca na linha 16.

Verifique no console que os valores do atributo marca são seus respectivos ids, 1 e 2. Isso ocorre, pois no banco de dados, armazenamos apenas uma referência à chave primária da entidade Marca.

E se quisermos substituir o id das marcas pelos seus respectivos nomes?

Para isso, precisamos realizar uma junção das tabelas Veiculo e Marca no comando SELECT.

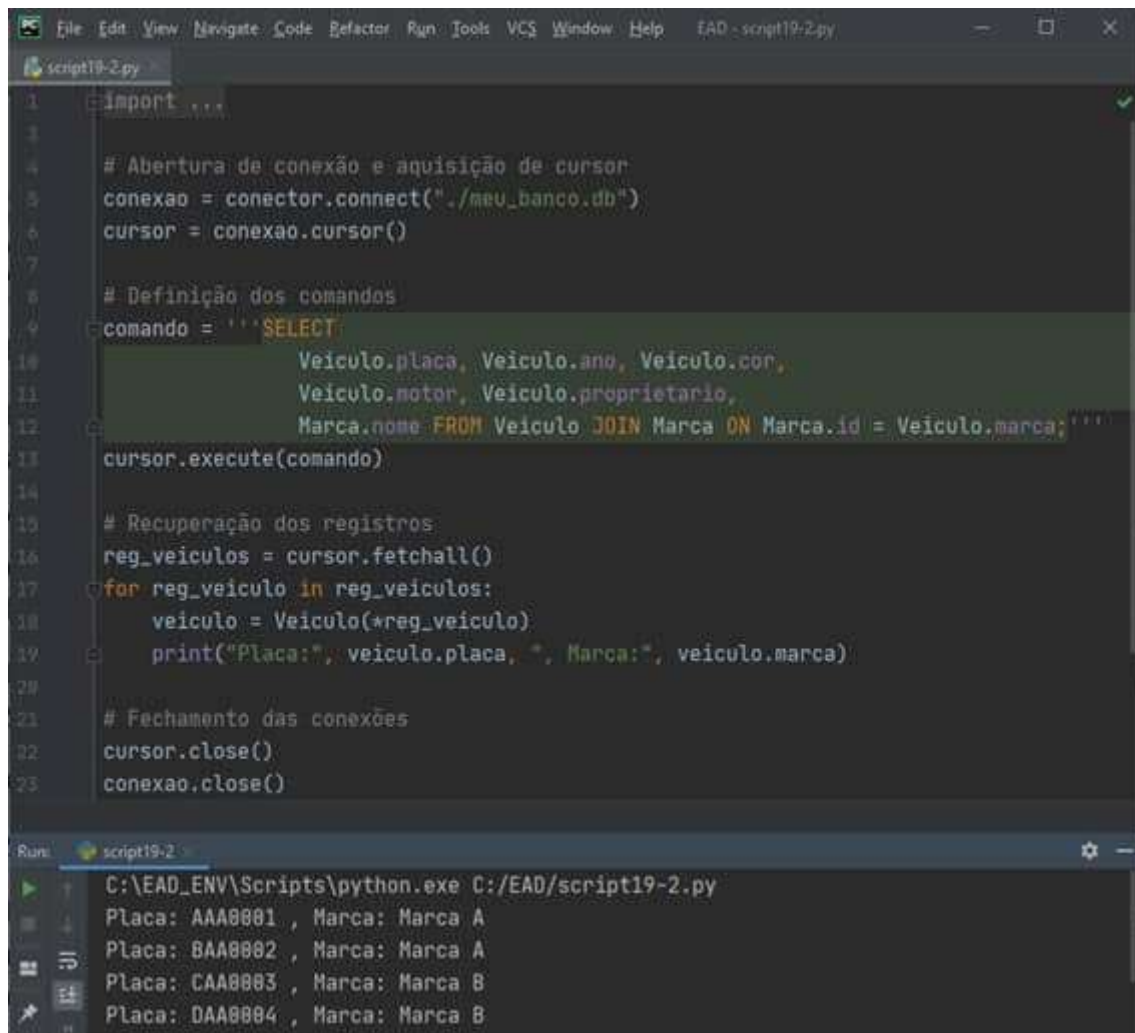
O comando SELECT para junção de duas tabelas tem a seguinte sintaxe:

***SELECT tab1.col1, tab1.col2, tab2.col1... FROM tab1 JOIN tab2 ON tab1.colN = tab2.colM;***

Primeiro, definimos quais colunas serão retornadas utilizando a sintaxe nome\_tabela.nome\_coluna, depois indicamos as tabelas que desejamos juntar e, por

último, indicamos como alinhar os registros de cada tabela, ou seja, quais são os atributos que devem ser iguais (colN e colM).

No exemplo a seguir, Figura 24, vamos criar um script de forma que o Veiculo tenha acesso ao nome da Marca, não ao id.



```
1 import ...
2
3
4 # Abertura de conexão e aquisição de cursor
5 conexao = conector.connect("./meu_banco.db")
6 cursor = conexao.cursor()
7
8 # Definição dos comandos
9 comando = '''SELECT
10     Veiculo.placa, Veiculo.ano, Veiculo.cor,
11     Veiculo.motor, Veiculo.proprietario,
12     Marca.nome FROM Veiculo JOIN Marca ON Marca.id = Veiculo.marca;'''
13 cursor.execute(comando)
14
15 # Recuperação dos registros
16 reg_veiculos = cursor.fetchall()
17 for reg_veiculo in reg_veiculos:
18     veiculo = Veiculo(*reg_veiculo)
19     print("Placa:", veiculo.placa, ", Marca:", veiculo.marca)
20
21 # Fechamento das conexões
22 cursor.close()
23 conexao.close()
```

Run: script19-2

```
C:\EAD_ENV\Scripts\python.exe C:/EAD/script19-2.py
Placa: AAA0001 , Marca: Marca A
Placa: BAA0002 , Marca: Marca A
Placa: CAA0003 , Marca: Marca B
Placa: DAA0004 , Marca: Marca B
```

Figura: 24.

Após criar uma conexão e obter um cursor, definimos o comando SQL para recuperar os dados das tabelas Veiculo e Marca de forma conjunta.

Observe o comando SQL nas linhas 9 a 12 e destacado a seguir:

**SELECT Veiculo.placa, Veiculo.ano, Veiculo.cor, Veiculo.motor, Veiculo.proprietario, Marca.nome FROM Veiculo JOIN Marca ON Marca.id = Veiculo.marca;**

Vamos selecionar os atributos placa, ano, cor, motor e proprietário do Veiculo e juntar com o atributo nome da tabela Marca. Observe que não vamos utilizar o atributo id da Marca.



As tuplas retornadas serão similares à seguinte:

```
('AAA0001', 2001, 'Prata', 1.0, 10000000099, 'Marca A')
```

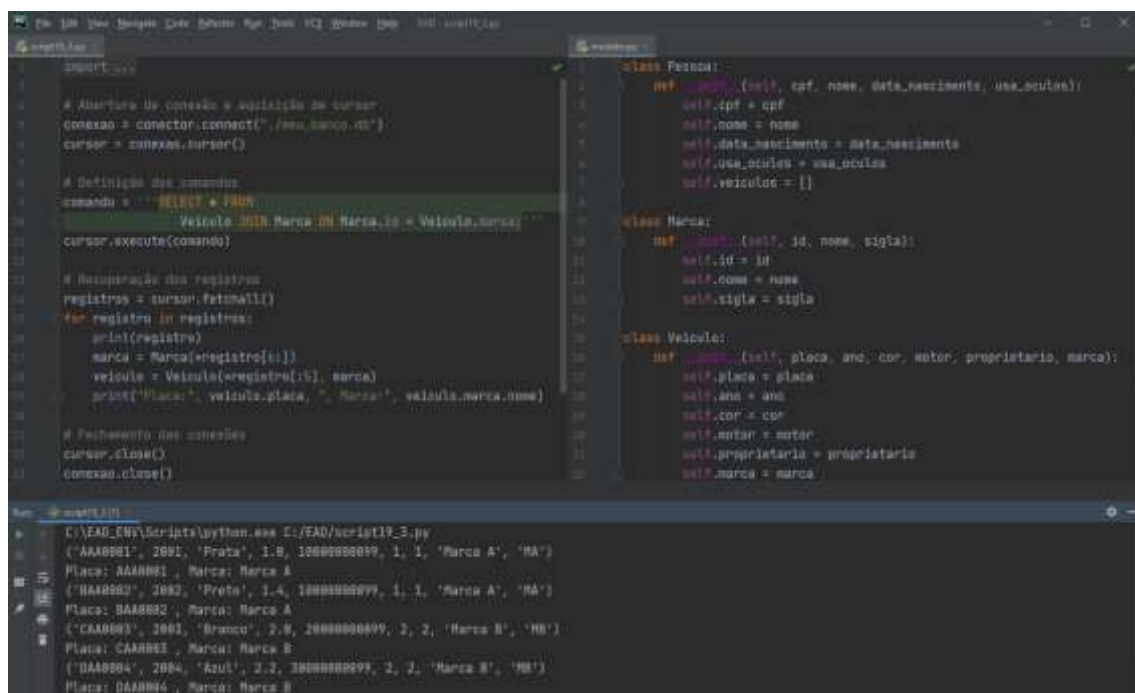
Na linha 16, recuperamos todos os registros de veículos, que foram iterados na linha 17.

Na linha 18, criamos um objeto do tipo Veiculo utilizando o operador \* e imprimimos os atributos placa e marca na linha 19.

Observe pelo console, que agora o atributo marca do nosso objeto do tipo Veiculo contém o nome da Marca.

No próximo exemplo, vamos dar um passo além. Vamos atribuir ao atributo marca, do Veiculo, um objeto do tipo Marca. Desta forma, vamos ter acesso a todos os atributos da marca de um veículo.

Para isso, vamos fazer alguns ajustes no nosso modelo. Observe a Figura 25 a seguir.



```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Abertura de conexão e aquisição de cursor
conexao = conector.connect("./src/banco.db")
cursor = conexao.cursor()

# Definição das queries
comando = """SELECT * FROM
            Veiculo JOIN Marca ON Marca.id = Veiculo.marca"""
cursor.execute(comando)

# Recuperação dos registros
registros = cursor.fetchall()
for registro in registros:
    print(registro)
    marca = Marca(*registro[6:])
    veiculo = Veiculo(*registro[:5], marca)
    print('Placa:', veiculo.placa, ", Marca:", veiculo.marca.nome)

# Fechamento das conexões
cursor.close()
conexao.close()
```

```
class Pessoa:
    def __init__(self, cpf, nome, data_nascimento, usa_oculos):
        self.cpf = cpf
        self.nome = nome
        self.data_nascimento = data_nascimento
        self.usa_oculos = usa_oculos
        self.veiculos = []

class Marca:
    def __init__(self, id, nome, sigla):
        self.id = id
        self.nome = nome
        self.sigla = sigla

class Veiculo:
    def __init__(self, placa, ano, cor, motor, proprietario, marca):
        self.placa = placa
        self.ano = ano
        self.cor = cor
        self.motor = motor
        self.proprietario = proprietario
        self.marca = marca
```

```
C:\EAD_CNF\Scripts\python.exe I:/EAD/scr19-3.py
('AAA0001', 2001, 'Prata', 1.0, 10000000099, 1, 1, 'Marca A', 'RA')
Placa: AAA0001 , Marca: Marca A
('AAA0002', 2002, 'Prata', 1.4, 10000000099, 1, 1, 'Marca A', 'RA')
Placa: AAA0002 , Marca: Marca A
('CAA0003', 2001, 'Branco', 2.0, 20000000099, 2, 2, 'Marca B', 'HB')
Placa: CAA0003 , Marca: Marca B
('DAA0004', 2004, 'Azul', 2.2, 30000000099, 2, 2, 'Marca B', 'HB')
Placa: DAA0004 , Marca: Marca B
```

Figura: 25.

Vamos começar pelo nosso modelo. Adicionamos o id ao construtor da classe Marca. Essa alteração foi feita para facilitar a criação do objeto do tipo Marca a partir da consulta no banco. Veremos o motivo mais à frente.

No script principal à esquerda da Figura 27, iniciamos o script com a abertura da conexão e criação do cursor.

Na sequência, na linha 9, criamos o comando **SQL SELECT** para retornar todas as colunas da tabela Veiculo e Marca, utilizando junção.

Na linha 11, executamos esse comando e os resultados da consulta foram recuperados na linha 14.

Na linha 15, iteramos sobre os registros recuperados.

Na linha 16, imprimimos cada tupla do registro da forma como foram retornados pelo conector. Vamos destacar um exemplo a seguir, que também pode ser observado no console.

```
('AAA0001', 2001, 'Prata', 1.0, 100000000099, 1, 1, 'Marca A', 'MA')
```

Destacado em vermelho, temos os atributos relacionados à entidade Veiculo e, em azul, temos os atributos relacionados à entidade Marca.

Na linha 17, utilizamos array slice para selecionar apenas os atributos da Marca. O resultado do slice para o exemplo anterior é a tupla (1, 'Marca A', 'MA'), onde temos os atributos id, nome e sigla. Essa tupla é utilizada em conjunto com o operador \* para criarmos um objeto do tipo Marca. Como agora temos acesso ao id da Marca, foi necessário adicionar o id ao construtor da classe Marca.

Para ilustrar, após o slice e desempacotamento, a linha 17 pode ser traduzida para o seguinte código:

```
marca = Marca(1, 'Marca A', 'MA')
```

Na linha 18, utilizamos array slice para selecionar apenas os atributos do Veiculo, que retornou a tupla ('AAA0001', 2001, 'Prata', 1.0, 100000000099), onde temos os atributos placa, ano, cor, motor e proprietário. Observe que removemos o id da Marca no slice. Fizemos isso, pois ele será substituído pelo objeto marca criado na linha 17.

Após o slice e desempacotamento, a linha 18 pode ser traduzida para o seguinte código:

```
veiculo = Veiculo('AAA0001', 2001, 'Prata', 1.0, 100000000099, marca)
```

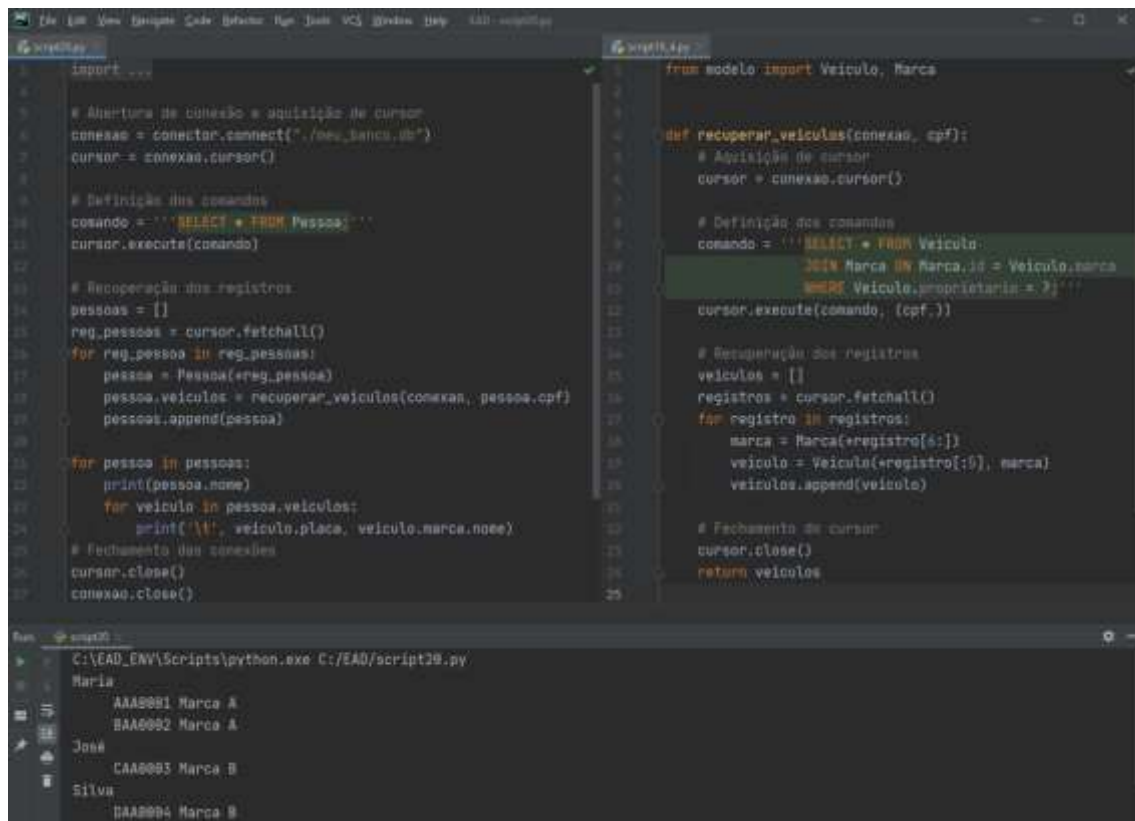
Na linha 19, imprimimos a placa do veículo e o nome da marca. Observe que o atributo marca, do Veiculo, faz uma referência ao objeto marca, por isso fomos capazes de acessar veiculo.marca.nome.

No final do script, fechamos a conexão e o cursor.

## SELEÇÃO DE REGISTROS RELACIONADOS

Para finalizar, vamos recuperar todas as pessoas, com seus respectivos veículos e marcas.

Para isso, vamos transformar o script anterior em uma função, de forma que possamos utilizá-la no nosso script final. Observe a Figura 26 a seguir.



```
import sys
import os

# Abertura de conexão e aquisição de cursor
conexao = connector.connect("./new_banco.db")
cursor = conexao.cursor()

# Definição dos comandos
comando = 'SELECT * FROM Pessoa;'
cursor.execute(comando)

# Recuperação dos registros
pessoas = []
reg_pessoas = cursor.fetchall()
for reg_pessoa in reg_pessoas:
    pessoa = Pessoa(*reg_pessoa)
    pessoa.veiculos = recuperar_veiculos(conexao, pessoa.cpf)
    pessoas.append(pessoa)

for pessoa in pessoas:
    print(pessoa.nome)
    for veiculo in pessoa.veiculos:
        print('\t', veiculo.placa, veiculo.marca.nome)

# Fechamento das conexões
cursor.close()
conexao.close()
```

```
from modelo import Veiculo, Marca

def recuperar_veiculos(conexao, cpf):
    # Aquisição de cursor
    cursor = conexao.cursor()

    # Definição dos comandos
    comando = 'SELECT * FROM Veiculo;'
    comando = comando + 'JOIN Marca ON Marca.id = Veiculo.marca'
    comando = comando + 'WHERE Veiculo.proprietario = ?;'
    cursor.execute(comando, (cpf,))

    # Recuperação dos registros
    veiculos = []
    registros = cursor.fetchall()
    for registro in registros:
        marca = Marca(*registro[:6])
        veiculo = Veiculo(*registro[6:], marca)
        veiculos.append(veiculo)

    # Fechamento do cursor
    cursor.close()
    return veiculos
```

```
C:\EAD_ENV\Scripts\python.exe C:/EAD/script20.py
Maria
AAAS001 Marca A
BAA0002 Marca A
João
CAAS003 Marca B
Silva
DAAS004 Marca B
```

Figura: 26.

À direita da imagem, temos o script 19\_4. Utilizamos como base o script do exemplo anterior, script19\_3, para criar uma função que retorne uma lista dos veículos de uma determinada pessoa.

Essa função tem como parâmetros uma conexão e o cpf de uma pessoa. Esse cpf será utilizado para filtrar os veículos que ela possui. Para isso, utilizamos o delimitador “?” na linha 11 e passamos o cpf da pessoa como argumento para o comando execute da linha 14.

Na linha 15, criamos uma lista vazia, chamada veiculos. Essa lista será povoada com os veículos recuperados pela consulta ao longo do laço for das linhas 17 a 20.

Ao final, fechamos o cursor e retornamos a lista veiculos.

À esquerda da figura, temos nosso script final, script20, cujo objetivo é ter uma lista com as pessoas cadastradas no banco de dados, incluindo seus veículos e marcas.

Após criar a conexão e o cursor, criamos o comando SQL para recuperar as pessoas na linha 10, que foi executado na linha 11.

Na linha 14, criamos a variável do tipo lista pessoas e, na linha 15, recuperamos todos os registros das pessoas utilizando a função fetchall do cursor.

Na linha 16, iteramos pelas pessoas recuperadas e para cada uma, criamos um objeto do tipo Pessoa (linha 17) e recuperamos seus veículos (linha 18).

Para recuperar os veículos, utilizamos a função recuperar\_veiculos do script 19\_4, passando como argumento a conexão criada na linha 6 e o cpf da pessoa.

Na linha 19, adicionamos cada pessoa à lista pessoas criada anteriormente.

Na linha 21, iteramos sobre a lista pessoas e imprimimos seu nome e a lista de veículos que possui.

Ao final, fechamos a conexão e o cursor.