

TUTORIAL: SQL - APLICANDO O CONHECIMENTO

MÓDULO I

VAMOS AO PASSO A PASSO - PARTE I:

A. Acesse o link <https://sqliteonline.com/>. Você será transferido para uma tela como a abaixo:

The screenshot shows the SQLiteOnline.com interface. At the top, there's a navigation bar with options like File, Run, Export, Import, and Client. On the left, a sidebar lists databases: SQLite (0.1.3 beta), MariaDB, PostgreSQL, and MS SQL. The main area shows a table named 'demo' with the following data:

ID	Name	Hint
1	Created by	A real human, without the use of artificial intelligence.
2	SQL Online	Next gen SQL editor
3	Kirill N	https://www.linkedin.com/in/sqliteonlinecom
4	Donate (ERC20: ETH or USDC)	0xCcc227E5615D4FADD758228Bab12ceb465D4ED18
5	Donate (BTC)	bctq25zqmglijzfz0lyduusylageh3jh7htcwjt2rdk
6	Chart	LINE-SELECT name, cos(id), sin(id) FROM demo;
7	Short CODE	s* tableName => SELECT * FROM tableName sf tableName => i\$SELECT @

On the right, there's a 'Syntax' panel with sections for Syntax and History, download links for remote databases and scripts, and links for all functions, comment, ALTER TABLE, ANALYZE, and ATTACH DATABASE.

B. Crie uma conta de usuário, para que scripts simples não sejam perdidos. Para isso, vá ao canto direito superior da tela, e clique em "sign in". Uma tela de pop-up se abrirá, e você poderá criar a sua conta.

The screenshot shows the 'Welcome to cloud .concept' login page. It features fields for E-mail and Password, and a 'Login' button. Below the form, a success message 'Success!' is displayed inside a Cloudflare frame. To the right, there's a list of actions: Create share link, Save cloud sql text (with a note about the 24 script, 8k char limit), and a note about logging after creation. At the bottom, there are links for Forgot password and Create an account, with a teal arrow pointing to the latter.

C. Agora, vamos aproveitar e já utilizar um comando em SQL: o Drop. Como este site é para pequenos experimentos e aprendizados, uma tabela chamada demo já está criada, como podemos ver a seguir. Como é necessário aplicar o conhecimento desde o zero, vamos deletar essa tabela. E nada melhor que utilizar o comando DROP. Lembrete: o comando DROP em SQL é utilizado para excluir um objeto do banco de dados, como uma tabela, índice, visão ou procedimento armazenado. Sendo assim, vamos à nossa tela de inserção de scripts. Na parte que está escrito SELECT * FROM demo; será onde escreveremos nossos comandos, tanto de inserção de dados quanto de criação de tabelas e consultas de registros.

The screenshot shows the SQLite interface. On the left, there's a sidebar with the following structure:

- SQLite (version 0.1.3 beta)
- Table
 - demo
- Column
 - ID INTEGER
 - Name varchar(20)
 - Hint TEXT

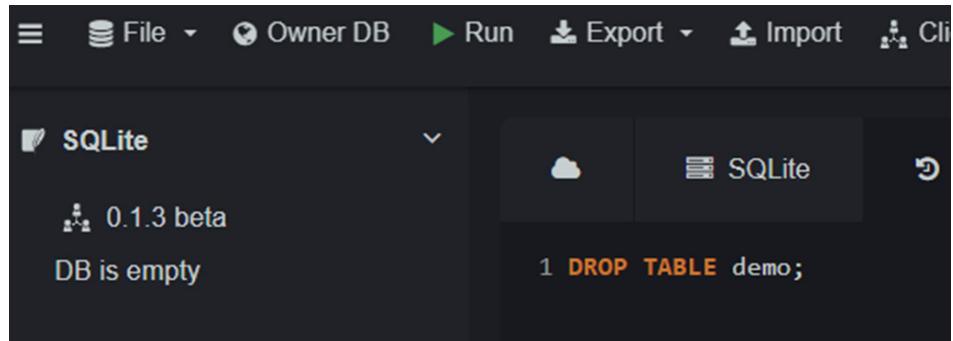
The main area shows a code editor with the following SQL query:1 SELECT * FROM demo;

D. Na parte logo abaixo, quando realizamos consultas, é possível ver os registros. Recorda-se que tínhamos os campos ID, NAME e HINT? Verifique na figura os registros que já tínhamos.

The screenshot shows the SQLite interface with the results of the query. The results are displayed in a table with the following data:

#	ID	Name	Hint
1		Created by	A real human, without the use of artificial intelligence.
2		SQL Online	Next gen SQL editor
3		Kirill N.	https://www.linkedin.com/in/sqliteonlinecom
4		Donate (ERC20, ETH or USDC)	0xCcc227E5615D4FADD758228Bab12ceb465D4ED18
5		Donate (BTC)	bc1q25zqmgll2fz0tyduusyfageh3jh7htcwjt2rdk
6		Chart	LINE-SELECT name, cos(id), sin(id) FROM demo;
7		Short CODE	s* tableName => SELECT * FROM tableName sf tableName => SELECT d

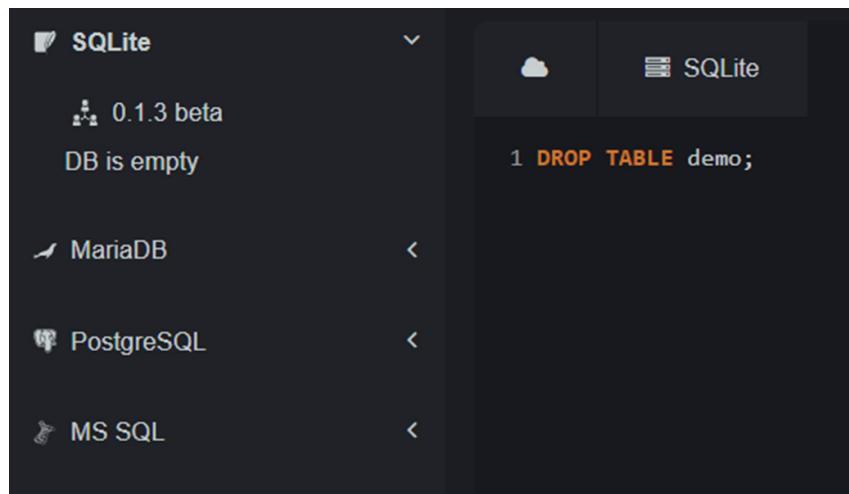
E. Agora, onde está escrito o comando de SELECT, apague e insira o comando DROP demo, devendo ficar como a figura abaixo mostra. Para o comando ser executado, precisamos clicar em RUN, no menu superior da nossa tela preta.



The screenshot shows a dark-themed SQLite database interface. At the top, there are standard menu items: File, Owner DB, Run, Export, Import, and Help. Below the menu, the connection status is shown as "SQLite 0.1.3 beta" and "DB is empty". In the main area, there is a single line of SQL code: "1 **DROP TABLE** demo;".

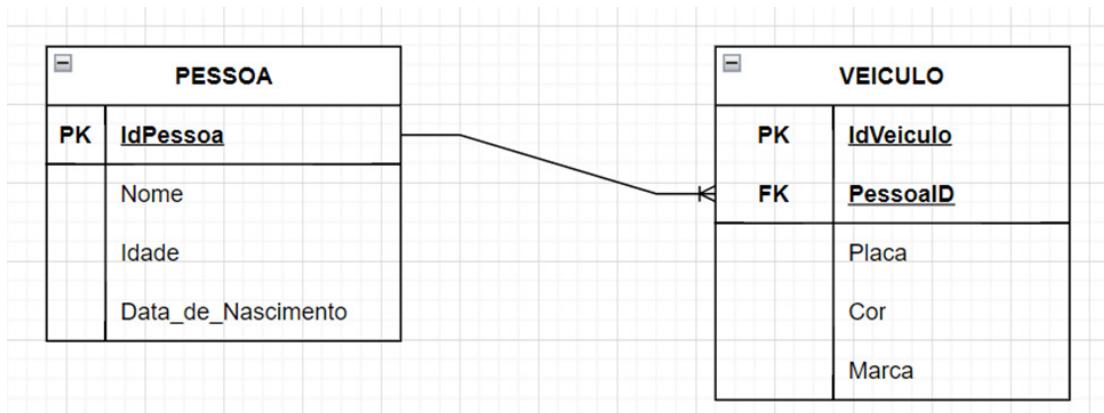
F. O DB está sem tabela alguma. Ótimo!

Primeiro comando seu executado com sucesso. Conseguimos realizar um **DROP** na tabela de demonstração do nosso pseudo sistema gerenciador de banco de dados. Disse pseudo porque esse sistema online possui cache temporário e é utilizado apenas para situações de aprendizado mesmo.



The screenshot shows a dark-themed SQLite database interface. On the left, there is a sidebar with a tree view showing "SQLite 0.1.3 beta", "DB is empty", and three other database entries: "MariaDB", "PostgreSQL", and "MS SQL". On the right, the main area shows the same SQL command as the previous screenshot: "1 **DROP TABLE** demo;".

Como disse anteriormente, você irá utilizar o diagrama entidade-relacionamento PESSOA-VEÍCULO que você criou anteriormente.



A primeira etapa que precisará realizar será a de transformarmos nossas entidades em tabelas reais no banco de dados.

VAMOS AO PASSO A PASSO – PARTE 2:

A. Continuando dos passos anteriores, utilize o comando CREATE para criar a tabela CLIENTE. A tabela possui os atributos NOME, IDADE e DATA_DE_NASCIMENTO. Ou seja, a primeira linha será CREATE TABLE pessoa. Aqui temos detalhes importantes:

- Os parênteses informam que dentro deles deverá haver entradas, que seriam quais colunas vamos inserir nas nossas tabelas.
- O ponto-e-vírgula no final é crucial. Esse ponto é muito esquecido pelos desenvolvedores iniciantes, e muitas vezes os compiladores mencionam que são outros erros, mas não o ponto-e-vírgula. Por isso, sugiro prestar atenção (mais ainda!).

```
1 CREATE TABLE pessoa(   
2  
3 );
```

B. Agora você irá inserir as colunas, ou seja, os atributos da nossa entidade pessoa, que está para se tornar uma tabela de banco de dados. Para isso, peço que copie o código abaixo na sua tela de script, do site que está aberto, e que a seguir vou lhe explicar:

```
1 CREATE TABLE PESSOA (   
2     nome VARCHAR(100),  
3     idade INT,  
4     data_de_nascimento DATE  
5 );
```

C. Todo atributo necessita ser catalogado como um tipo. Como parâmetros de entrada temos:

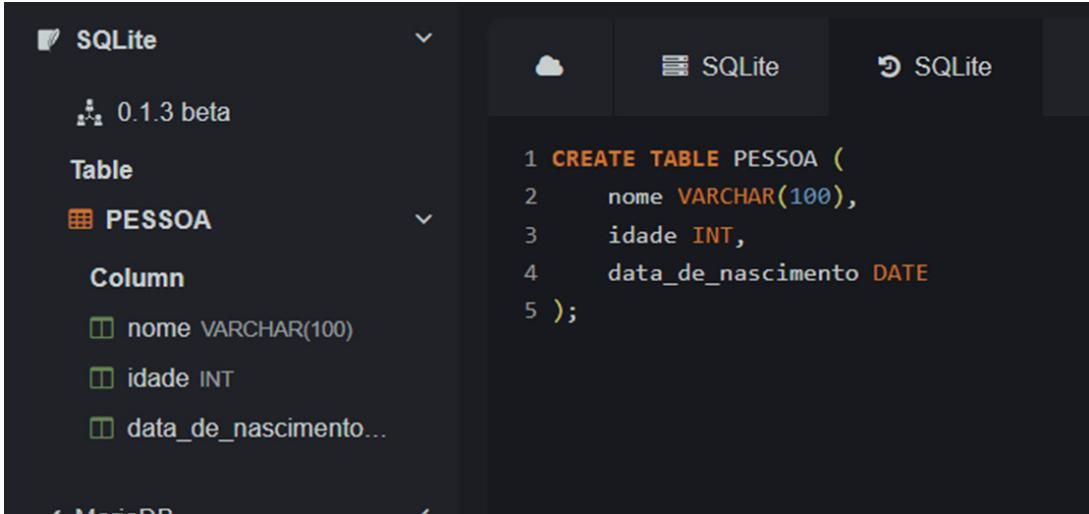
- Nome, que é um texto, certo? No SQL chamamos de VARCHAR. E necessitamos colocar um número de caracteres máximo para esse tipo, para evitar utilização excessiva de memória. Lembre-se, de seus conhecimentos anteriores, alocar memória e não a utilizar faz com que seus programas ou aplicações se tornem lentos e inviáveis de serem utilizados.
- Idade, que basicamente é um numeral. Como não trabalharemos com números racionais, apenas inteiros, vamos declarar esse atributo como INT.

- Por último, mas não menos importante, temos DATA_DE_NASCIMENTO, em que temos um tipo específico chamado DATE.

Como já feito anteriormente, reavale se o código foi escrito corretamente e clique em RUN, na sua tela do site em que você está rodando o SQL script.

D. IMPORTANTE: caso você já tenha conhecimento em SQL e em IDEs (programas que utilizamos para gerenciar os bancos de dados), pode utilizá-los também. Fica a seu critério!

Feito isso, tem-se de resultado a tabela PESSOA criada:



The screenshot shows the SQLite interface with the following details:

- SQLite Version:** 0.1.3 beta
- Table:** PESSOA
- Columns:**
 - nome: VARCHAR(100)
 - idade: INT
 - data_de_nascimento: DATE
- SQL Script:**

```

1 CREATE TABLE PESSOA (
2     nome VARCHAR(100),
3     idade INT,
4     data_de_nascimento DATE
5 );

```

E. Antes de inserir registros nessa tabela, você se recorda do nosso diagrama entidade-relacionamento, em que havia um campo de identificação chamado IdPessoa?

Não se preocupe. Não pedi para você inserir de propósito. Ele é de extrema importância para garantirmos a integridade dos relacionamentos do nosso banco de dados. Não é interesse seu que os VEÍCULOS sejam vinculados a mais de uma pessoa, correto?

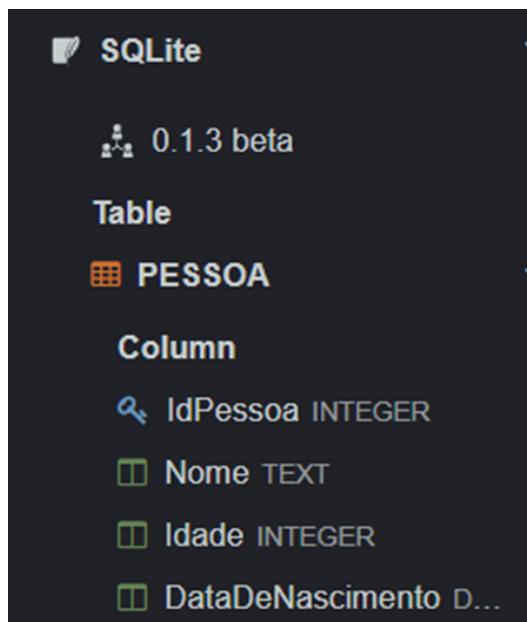
Deve-se respeitar o conceito de **imutabilidade** – uma vez definida, a chave primária não deve ser alterada. Isso garante a integridade dos dados e a consistência das relações entre as tabelas.

Então, vamos utilizar o DROP novamente nessa tabela e vamos criar agora com a chave primária!

F. O SQLite que você está utilizando agora não consegue fazer a alteração diretamente. Mais adiante vamos trabalhar com SGDBs mais poderosos que lhe deixarão fazer isso. Com o código abaixo, conseguimos deletar a tabela pessoa, que não possuía chave primária. Segundo ponto, criaremos novamente a tabela pessoa, agora com IdPessoa sendo um número inteiro, e agora definido como PRIMARY KEY, ou seja, chave primária, além de ser auto-increment, que fará com que se inicie a contagem em 1, e não seja necessário um controle das próximas inserções de registros na tabela pessoa.

```
1 -- Drop tabela pessoa se existir
2 DROP TABLE IF EXISTS pessoa;
3
4 -- Criação da tabela pessoa com a nova estrutura
5 CREATE TABLE PESSOA (
6     IdPessoa INTEGER PRIMARY KEY AUTOINCREMENT,
7     Nome TEXT,
8     Idade INTEGER,
9     DataDeNascimento DATE
10 );
```

G. Como resultado, temos a tabela PESSOA criada. Agora podemos inserir registros do tipo da entidade PESSOA em nossa tabela.



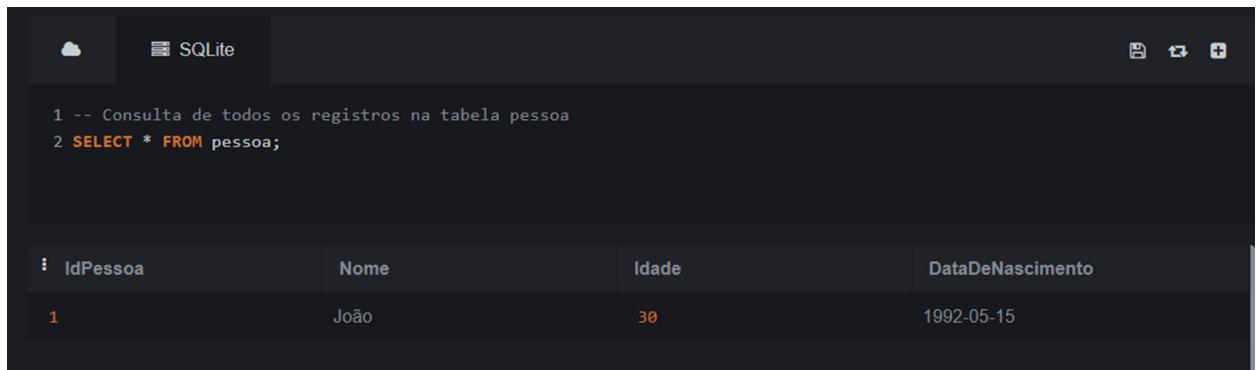
H. Como será feito isso? Utilizaremos os comandos INSERT. A operação INSERT serve para inserirmos registros em nossa tabela. O conector INTO relaciona qual tabela, no caso PESSOA.

Importante: quando definimos um IdPessoa como auto-increment, ele não precisa vir como atribuíto, pois será definido automaticamente. Agora você pode consultar todos os registros inseridos na sua tabela PESSOA.

```
1 -- Inserção de um registro na tabela pessoa
2 INSERT INTO pessoa (Nome, Idade, DataDeNascimento)
3 VALUES ('João', 30, '1992-05-15');
```

I. Execute o comando SELECT e vamos explicar detalhadamente o que ocorre:

- O comando SELECT realiza consultas em sua tabela.
- Quando citamos * , significa que queremos todos os registros existentes.
- From traduzimos literalmente do inglês: qual será a origem?
- E por último, qual tabela, no caso, PESSOA.



The screenshot shows the SQLite Database Browser interface. At the top, there's a toolbar with icons for file operations. Below it, the title bar says "SQLite". The main area has a dark background with white text. A SQL query is displayed:

```
1 -- Consulta de todos os registros na tabela pessoa
2 SELECT * FROM pessoa;
```

Below the query, a table is shown with four columns: IdPessoa, Nome, Idade, and DataDeNascimento. There is one row of data:

	IdPessoa	Nome	Idade	DataDeNascimento
1	1	João	30	1992-05-15

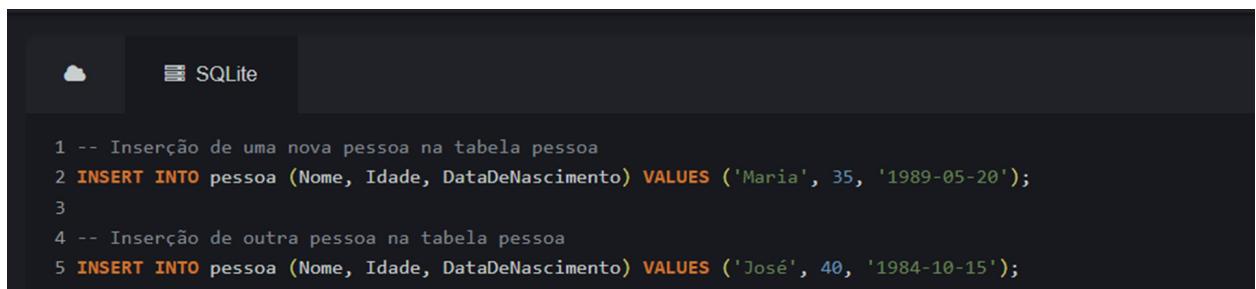
J. No prompt inferior, aparecem as colunas IdPessoa, Nome, Idade e DataDeNascimento e abaixo todos os registros, divididos por linhas.

Parabéns! Você atingiu uma etapa fundamental aqui no seu curso de Desenvolvimento Back-End. Criou diagramas, tabelas e realizou consultas! Ótimo!

Agora, vamos inserir mais duas pessoas?

PASSO A PASSO – PARTE 3:

A. Se fizer a consulta de SELECT * FROM Pessoa, você obterá:



The screenshot shows the SQLite Database Browser interface. At the top, there's a toolbar with icons for file operations. Below it, the title bar says "SQLite". The main area has a dark background with white text. A SQL query is displayed:

```
1 -- Inserção de uma nova pessoa na tabela pessoa
2 INSERT INTO pessoa (Nome, Idade, DataDeNascimento) VALUES ('Maria', 35, '1989-05-20');
3
4 -- Inserção de outra pessoa na tabela pessoa
5 INSERT INTO pessoa (Nome, Idade, DataDeNascimento) VALUES ('José', 40, '1984-10-15');
```

B. E se você quiser apenas o segundo da lista, que seria a pessoa com nome Maria?

SQLite

```
1 -- Consulta de todos os registros na tabela pessoa
2 SELECT * FROM pessoa;
```

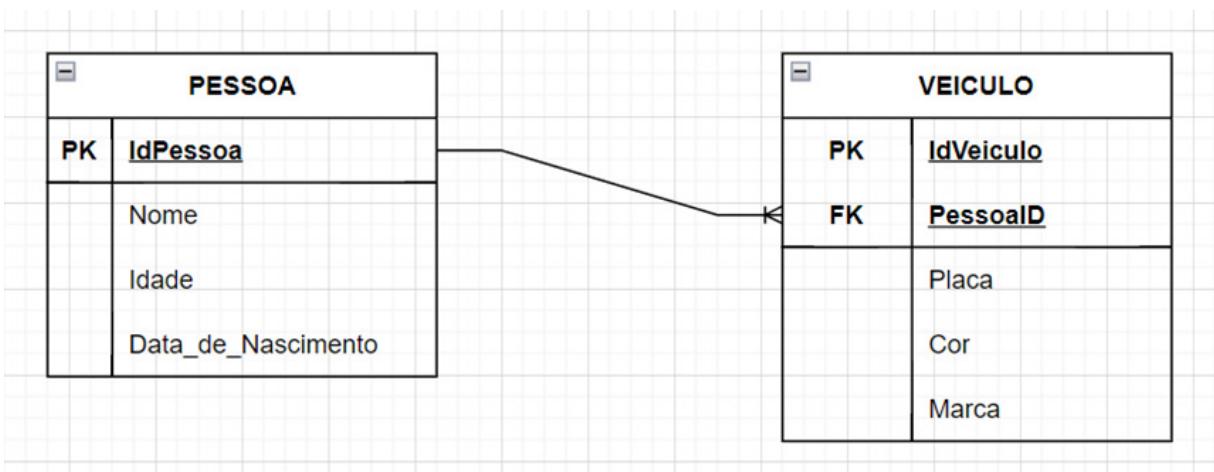
IdPessoa	Nome	Idade	DataDeNascimento
1	João	30	1992-05-15
2	Maria	35	1989-05-20
3	José	40	1984-10-15

C. Então, se insere a cláusula WHERE, informando qual Pessoa relacionada a uma específica IdPessoa você quer que apareça. Nesse caso, 2.

```
1 -- Consulta de todos os registros na tabela pessoa
2 SELECT * FROM pessoa WHERE IdPessoa = 2;
```

IdPessoa	Nome	Idade	DataDeNascimento
2	Maria	35	1989-05-20

D. Agora, para continuar nossa reflexão, precisará fazer o mesmo com a tabela veículos, seguindo nosso diagrama:



E. Para isso, teremos que criar uma tabela VEICULO, com a chave primária IdVeiculo, a chave estrangeira PessoalD, os atributos Placa, Cor e Marca. Para entender o script abaixo, a primeira linha, como já explicado, será a chave primária, IdVeiculo. Nossa chave exclusiva para cada veículo, do tipo inteiro e auto incrementável. Temos Placa, Cor, Modelo como texto – outro tipo que podemos inserir ao invés de VARCHAR.

```
1 -- Criação da tabela Veiculo
2 CREATE TABLE VEICULO (
3     IdVeiculo INTEGER PRIMARY KEY AUTOINCREMENT,
4     Placa TEXT,
5     Cor TEXT,
6     Modelo TEXT,
7     PessoaId INTEGER,
8     FOREIGN KEY (PessoaId) REFERENCES Pessoa(IdPessoa)
9 );
10
```

F. Por último, criamos o atributo PessoalD como inteiro também. A diferença é que posteriormente definimos essa mesma PessoalD como chave estrangeira, como está escrito FOREIGN KEY.

Referenciando esta, utilizando a cláusula REFERENCES Pessoa (IdPessoa). Ou seja, vinculando com a tabela Pessoa, e exclusivamente a chave primária IdPessoa desta tabela.

G. Para entender melhor a mágica, sugiro inserir alguns registros, como no exemplo abaixo. Pode-se inserir vários registros em um só INSERT. Porém, lembre-se que precisamos sempre inserir qual ID vamos referenciar na tabela IdPessoa.

```
1 -- Inserção de 6 veículos vinculados aos 3 clientes existentes
2 INSERT INTO Veiculo (Placa, Cor, Modelo, PessoaId)
3 VALUES
4     ('ABC123', 'Preto', 'Sedan', 1),    -- Pessoa 1
5     ('XYZ456', 'Branco', 'SUV', 2),      -- Pessoa 2
6     ('DEF789', 'Prata', 'Hatchback', 3),  -- Pessoa 3
7     ('GHI123', 'Azul', 'Picape', 1),     -- Pessoa 1
8     ('JKL456', 'Vermelho', 'Caminhonete', 2), -- Pessoa 2
9     ('MNO789', 'Verde', 'Compacto', 3);   -- Pessoa 3
```

H. Se você fizer um SELECT * FROM Veiculo, você terá o seguinte resultado:

The screenshot shows a SQLite database interface with a single table named 'VEICULO'. The table has columns: IdVeiculo, Placa, Cor, Modelo, and Pessoaid. There are 6 rows of data:

	IdVeiculo	Placa	Cor	Modelo	Pessoaid
1	1	ABC123	Preto	Sedan	1
2	2	XYZ456	Branco	SUV	2
3	3	DEF789	Prata	Hatchback	3
4	4	GHI123	Azul	Picape	1
5	5	JKL456	Vermelho	Caminhonete	2
6	6	MNO789	Verde	Compacto	3

I. Não confunda o IdVeiculo com a Pessoaid. Faça um select para ver quantos carros o IdPessoa “1” tem:

The screenshot shows a SQLite database interface with a single table named 'VEICULO'. The query executed is 'SELECT * FROM VEICULO WHERE Pessoaid = 1'. There are 2 rows of data:

	IdVeiculo	Placa	Cor	Modelo	Pessoaid
1	1	ABC123	Preto	Sedan	1
4	4	GHI123	Azul	Picape	1

J. A pessoa com Pessoaid “1” tem dois veículos. Mas você sabe qual é essa pessoa? Então precisaremos utilizar uma cláusula JOIN, que faz com que se consiga juntar duas tabelas e utilizar a cláusula de filtro WHERE:

The screenshot shows a SQLite database interface with two tables: 'Pessoa' and 'VEICULO'. The query executed is:

```
1 SELECT Pessoa.Nome, Veiculo.Placa, Veiculo.Cor, Veiculo.Modelo
2 FROM Pessoa
3 JOIN Veiculo ON Pessoa.IdPessoa = Veiculo.Pessoaid
4 WHERE Pessoa.IdPessoa = 1;
```

The results show the names of the people and the details of their vehicles:

	Nome	Placa	Cor	Modelo
1	João	ABC123	Preto	Sedan
2	João	GHI123	Azul	Picape

K. Entenda por partes o que foi feito:

- Pessoa.Nome seleciona o nome da pessoa.
- Veiculo.Placa, Veiculo.Cor e Veiculo.Modelo selecionam os atributos do veículo.
- JOIN Pessoa ON Veiculo.Pessoald = Pessoa.IdPessoa combina as tabelas “Pessoa” e “Veiculo” usando a chave estrangeira “Pessoald” na tabela “Veiculo” e a chave primária “IdPessoa” na tabela “Pessoa”.
- WHERE Pessoa.IdPessoa = 1 filtra os resultados apenas para a Pessoa com ID igual a 1 (que é a Pessoa 1).