SQL para Crianças

**━━━━━━━━━━━━━━━━━━━━━━━**

Sumário

[Motivo de Aprender SQL 2](#_Toc79682942)

[Como Manipular esses Dados 3](#_Toc79682943)

[Significado da sigla SQL 3](#_Toc79682944)

[Qual Ferramenta Usar? 4](#_Toc79682945)

[SQL Fundamentos dos Dados 5](#_Toc79682946)

[Tipos de dados Literais e Caracteres 5](#_Toc79682947)

[Tipos de dados Numéricos 6](#_Toc79682948)

[Tipos de dados Temporais 7](#_Toc79682949)

[Tipos de dados Booleanos 8](#_Toc79682950)

[Palavras Reservadas 8](#_Toc79682951)

[Conceito de Chaves 9](#_Toc79682952)

[Operadores de relacionais e Matemáticos 9](#_Toc79682953)

[Operadores Lógicos 10](#_Toc79682954)

[SQLite Funcionalidades 10](#_Toc79682955)

[Interface do SQLite 11](#_Toc79682956)

[Primeira Base de Dados 12](#_Toc79682957)

[Consultas Básicas 15](#_Toc79682958)

[Sintax e Select 15](#_Toc79682959)

[Consultas com Where 18](#_Toc79682960)

[Trabalhando com Ordenação 20](#_Toc79682961)

[Instruções com Tabelas 21](#_Toc79682962)

[SQL Server Funcionalidades 24](#_Toc79682963)

[Fundamentos das Tabelas 24](#_Toc79682964)

[Palavras Reservadas para Tabelas 25](#_Toc79682965)

[Alterar e modificar colunas 27](#_Toc79682966)

[Funções e Apelidos 29](#_Toc79682967)

[Instruções de Agrupamento 31](#_Toc79682968)

[Sub Consulta 32](#_Toc79682969)

[Junções 33](#_Toc79682970)

[Restaurando uma base de dados 37](#_Toc79682971)

[Glossário 40](#_Toc79682972)

[Bibliografia 42](#_Toc79682973)

[Informações Adicionais 43](#_Toc79682974)

**Objetivo**

Resumir a linguagem SQL para leigos, aprender com exemplos simples, rápidos e com alguns exercícios, na verdade vários!

**Resumo**

Esse mini guia está dividido em três partes.

1. Introdução: Parte Introdutória, o capítulo um, nesse capitulo será citado um pouco sobre SQL, como ela funciona e download do SQLite. Recomendo acessar o Glossário no final desseguia para entender um pouco mais dos termos que irei utilizar nesse guia. *Talvez eu venha futuramente complementar com NoSQL como o MongoDB*.
2. Primeiros Comandos: No capítulo dois, será a parte introdutória as primeiras consultas e criar a primeira Base de Dados tudo com um software leve, grátis e móvel, o SQLite.
3. Softwares Profissionais: Não que o SQlite não seja um software poderoso, você vai entender ainda nesse capítulo um, o motivo de usar o SQLite, mas, também durante o decorrer desse mini guia, será utilizado um software novo, o SQL Server, o terceiro software mais utilizado no mercado.

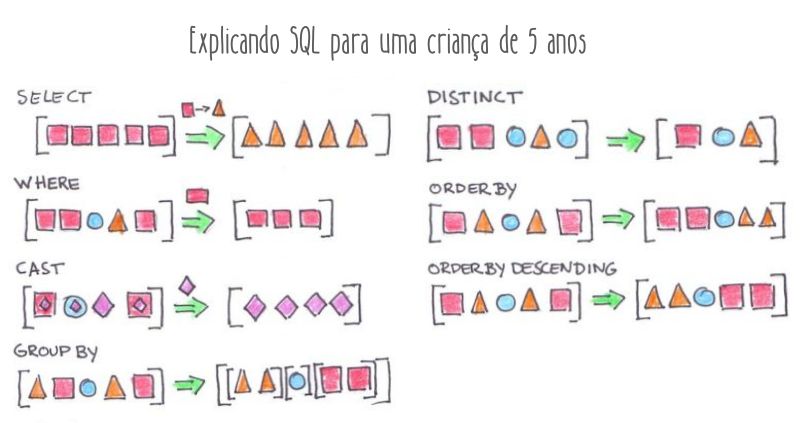
# Motivo de Aprender SQL

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

SQL é uma das poucas coisas desse mundo que consegue chegar próximo à unanimidade, desenvolvida na década de 1970 por funcionários da atual IBM, e até hoje é utilizada em qualquer Banco de Dados Relacional.

Qualquer estabelecimento, escola e até empresas, possuem e armazenam seus dados de várias formas, mas, todas armazenam.

## Como Manipular esses Dados



Para conseguir manipular esses dados, usamos um SGBD*,* com esse software para gerenciar nossas bases de dados, temos inúmeras vantagens, como:

* Visualização interativa dos dados.
* Compartilhamento dos dados.
* Backup, segurança e recuperação de falhas.
* Flexibilidade e padronização.

## Significado da sigla SQL

Significa "Structured Query Language”, ou “Linguagem de Consulta Estruturada". É uma linguagem para a manipulação de uma Base de Dados Relacional.

Usamos diversos comandos, que são denominados como instruções, as instruções retornam à consulta, os dados da minha base de dados. Para realizar uma consulta, antes temos que entender as instruções para realizar essas determinadas ações, as instruções estão divididas em três principais grupos:

1. DML (Data Manipulation Language).

* SELECT -> Pesquisa.
* UPDATE -> Atualização.
* DELETE -> Deletar.
* INSERT -> Inserir.

1. DDL (Data Definition Language).

* CREATE -> Criar.
* ALTER -> Alterar.
* DROP -> Deletar.

1. DCL (Data Control Language).

* GRAND -> Fornecer Privilégios.
* REVOKE -> Remover Privilégios. (Ex: Acesso)

1. DTL (Data Transaction Language).

* COMMIT -> Salvar.
* ROLLBACK -> Voltar ao ponto antes de inserir os registros.

Usamos os comandos DLL, geralmente em Objetos (Tabelas).

## Qual Ferramenta Usar?

Existem inúmeras ferramentas para manipular uma base de dados com uma interface gráfica.

A mais simples e leve é o SQLite, ser simples não quer dizer que é ruim ou muito básica, e sim, ser fácil de usar, extremamente leve, esse sistema é usado nos smartphones, não requer instalação, registro de root, etc. Baixou, funcionou.

Link para o Download do SQLite Browser: <https://sqlitebrowser.org/dl/>

Windows: Basta baixar a versão conforme o tipo do seu sistema.

Ubuntu: Abra o terminal, para abrir o terminal (Ctrl + Alt + T) e digite *sudo apt-get install sqlitebrowser*, mas antes, uma boa pratica sempre que abrir o terminal, no Ubuntu é atualizar o gerenciador de pacotes usando o comando *sudo apt-get update.* Caso necessário do uso do Personal Package, use o comando *sudo add-apt-repository -y ppa:linuxgndu/sqlitebrowser.*

Mac: Com o homebrew fica tudo mais fácil, caso precise instalar o mesmo, no próprio site oficial do brew tem a sua instalação, <https://docs.brew.sh/Installation> .

*brew install --cask db-browser-for-sqlit*.

# SQL Fundamentos dos Dados

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

A Linguagem SQL é como outra linguagem, tem seus tipos de dados e formas específicas para tais manipulações, um exemplo para citar esses tipos é que não podemos colocar uma sequência de caracteres textuais em uma coluna que só aceita números.

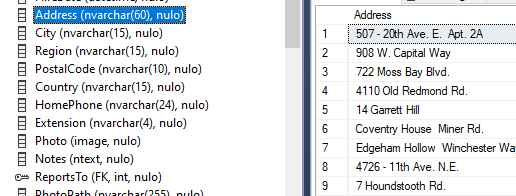
Você precisa especificar na criação de uma tabela, os tipos de dado que serão aceitos durante a manipulação do banco de dados, na linguagem SQL, existem quatro famílias principais desses tipos de dados.

## Tipos de dados Literais e Caracteres

Para qualquer linguagem de programação, no contexto de programação de computadores, o conjunto de caracteres é denominado ***String***, utilizada para representar palavras, frases e textos. Os tipos de dados em texto, sempre são representados com aspas simples ou duplas.

Nome: ‘Zacarias, Sobrenome: ‘Junior.

Para o SQL, podendo ter valores numerais e símbolos incluídos em uma sequência de caracteres que podem ser armazenados em variáveis.



Repare na coluna *Address*, contem letras, símbolos e números, mais seu tipo de dado é *nvarchar*. Que é um tipo de variável no SQL para *string*, o outro valor dentro dos parênteses, representa ‘nulo’, instruções geralmente dentro de parênteses, denominamos parâmetros, mas não quer dizer que os valores ali são nulos, veremos nos capítulos seguintes.

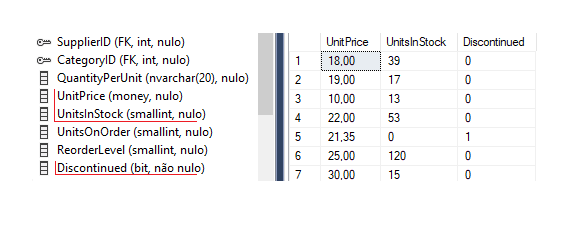
Mas não é só esses valores que existem.

|  |  |
| --- | --- |
| Char | Variável que armazena letras, números e caracteres especiais, contém um valor fixo que é um parâmetro, caso não seja preenchido com a sequência de caracteres, o SQL preenchera com espaços em branco. |
| Varchar e Nvarchar | Mesmo conceito anterior, porém, é flexível em relação aos valores armazenados, ou seja, caso o parâmetro for (100), e os valores armazenados não alcançarem um total de 100 caracteres, a variável vai adequar-se ao tamanho dos valores inseridos. |
| TinyText e LongText | Tipo de dado somente texto, varia entre TinyText, Text, MediumText e LongText, a sua diferença é a quantidade de caracteres que você quer armazenar. |
| TinyBlob e LongBlob | Valores Binários como arquivos, fotos, etc. o tamanho também varia como os estilos textuais. |
| Set e Enum | Ambos são para uma lista de valores, contudo, o Enum é uma lista mais robusta e armazena mais valores. |

Esses são todos os tipos de dados para valores caracteres e literais. Durante a criação de uma tabela no próximo capitulo, vai ser especificado esses valores.

## Tipos de dados Numéricos

Durante a criação das colunas e das tabelas, o tipo de dado numérico é bem importante, com esse tipo de dado, podemos armazenas dados de vendas, realizar contas matemáticas entre outras coisas.



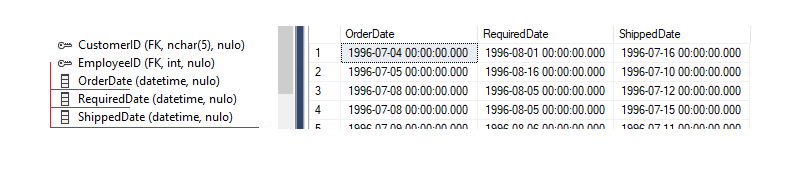
Nessa seleção de colunas, temos três tipos de valores numéricos veja mais alguns.

O parâmetro ‘Nulo’, não está relacionado ao valor numérico para ser nulo, é um outro conceito que será abordado ao decorrer desse capitulo.

|  |  |
| --- | --- |
| Bit | Número inteiro de 0 ou 1 e nulo. |
| Int | É subdividido em TinyInt, SmallInt, Int e BigInt, o que muda é o espaço para armazenar o conjunto numérico. |
| Decimal | Permite valores que serão convertidos, você especifica as escalas como parâmetros.  Valores exatos com partes fracionárias. |
| Numeric | Mesmo conceito do decimal, contudo, não possui valor depois do ponto. (Precisão, Escala), Exemplo, (2, 5), 20,12753. |
| Money | Formato de dados para valores monetários. |
| Float | Valores com ponto flutuante, seguem um padrão de arquitetura (IEEE 754). |
| Real | Mesmo conceito do *float*, a variação é a precisão. |

## Tipos de dados Temporais

Um tipo dedado especialmente desenvolvido para armazenar valores de datas e tempos.



Reparem que mesmo em uma base de dados estrangeira, o formato de datas é o mesmo, sendo primeiro ano, depois mês e dia, com os horários separados por dois pontos.

|  |  |
| --- | --- |
| Time | Armazena somente os valores de Tempo. |
| Datetime | Armazena o formato da data em ( AAAA-MM-DD HH:MM:SS ) Ano-Mês-Dia Hora:Minuto:Segundo |
| Datetime2 | Armazena o formato da data em ( AAAA-MM-DD HH:MM:SS:MS ) Ano-Mês-Dia Hora:Minuto:Segundo:Milisegundo |
| SmallDatetime | Armazena o formato de Datetime, a partir de um limite de datas estabelecidos pelo próprio SQL. |
| Date | Armazena o formato da data em ( AAAA-MM-DD ) sem os horários. |
| Datetimeoffset | Mesma coisa que o Datatime2, porem com adição do Fuso horário |

## Tipos de dados Booleanos

Esse é o tipo de dado mais fácil, é como o BIT, ele armazena apenas dois valores, 1, 0 ou nulo. Onde o valor zero é considerado como falso e o valor um é considerado verdadeiro. Exemplo, ‘Na minha base de dados, os valores da coluna *casado*, está com valores um ou zero, ou seja, sim, está casado ou, não está casado’.

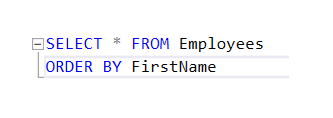
## Palavras Reservadas

Em toda a linguagem SQL ou até mesmo em outras linguagens de programação como Java Script, Python, possuem suas palavras reservadas, ou seja, palavras que usamos para realizar nossas instruções, no exemplo do Java Script, não podemos utilizar a palavra ‘Let’, pois é uma palavra reservada da linguagem para indicar que nas próximas sequencias da linha, tem uma variável.

Em SQL, existem três formas de diferencias as palavras reservadas, são elas, a coloração que o SGBD nos indica quando digitamos os comandos no software, ele não é case sensitive, mas é possível escrever nossas instruções em letra maiúscula e todas as palavras são, a maioria auto explicativas.

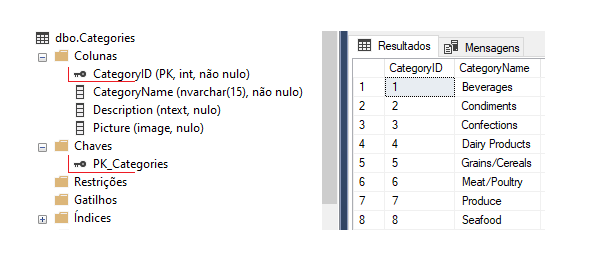
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CREATE | DATABASE | SELECT | FROM | WHERE | DROP | DELETE | UPDATE |
| Criar | Base de Dados | Selecionar | A partir de | Onde | Largar / Jogar | Deletar | Atualizar |

Esses são alguns exemplos de palavras reservadas da linguagem SQL (As que estão em Negrito), existem muitas outras palavras reservadas que irão aparecer durante as consultas.

Exemplo de uma Instrução.

## Conceito de Chaves

Durante a criação das nossas tabelas, todas as tabelas devem possuir uma chave, denominada Id, geralmente essa chave tem o mesmo nome da Tabela, juntamente com a palavra Id no final.



Reparem que a chave até recebe um apelido de PK, essa chave é o indicador dessa tabela para as outras tabelas.

Quando uma chave primária está pertencendo a uma linha de outra tabela, ou seja, existe a tabela X que possui sua chave primária, porem ela aparece na tabela Y, ou seja, a tabela Y usa e referência a tabela X, quando a PK de uma tabela aparece em outra tabela, ela é chamada de *Foreign Key*.

Observação, a PK, não pode ter um valor nulo, os valores da coluna do Id sempre têm que serem sempre únicos, pois são os Index os valores tem que ser incrementais e sempre começam com o valor 1.

## Operadores de relacionais e Matemáticos

A linguagem SQL suporta alguns valores matemáticos, mas os principais são as condicionais, que usaremos no próximo capitulo.

|  |  |
| --- | --- |
| Igualdade  = | Igualdade, essa condicional vai verificar se o valor X é igual ao valor Y ( X = Y ), nesses casos, retorna uma condição booleana (Sim ou Não).  ( ‘A’ = ‘A’ ), o resultado vai ser positivo. |
| Diferente  <> | Diferente, essa condicional vai verificar se o valor X é diferente do valor Y ( X <> Y ), nesses casos, retorna uma condição booleana (Sim ou Não).  ( 3 <> 1 ), o resultado vai ser positivo. |
| Maior Que  > | Maior que, essa condicional vai verificar se o valor X é maior do que valor Y ( X > Y ), nesses casos, retorna uma condição booleana (Sim ou Não).  ( 7 > 6 ), o resultado vai ser positivo. |
| Menor Que  < | Menor que, essa condicional vai verificar se o valor X é menor do que valor Y ( X < Y ), nesses casos, retorna uma condição booleana (Sim ou Não).  ( 10 < 100 ), o resultado vai ser positivo. |
| Maior e Igual  >= | Esse conceito se aplica também a o **menor ou igual**, ele verifica se o valor X é maior ou idêntico ao valor Y.  ( 10 <= 50 ) Positivo, ( 20 >= 20 ) Positivo. |
| Matemáticos | Adição ( + ), Subtração ( - ), Divisão ( / ), Multiplicação ( \* ).  Veremos mais adiante as próprias funções do SQL para aplicar outras funções matemáticas. |

## Operadores Lógicos

Diferente dos operadores relacionais, os operadores lógicos comparam resultados booleanos, (True ou False) os operadores relacionais, seu resultado é um valor booleano, com esse valor podemos utilizar operadores lógicos para ser ainda mais cauteloso na nossa consulta.

Os operadores lógicos são divididos em três.

|  |  |  |  |
| --- | --- | --- | --- |
| Expressões1 | Expressões2 | AND / E | OR / OU |
| Verdadeiro | Verdadeiro | Verdadeiro | Verdadeiro |
| Falso | Falso | Falso | Falso | **NOT / Não** |
| Falso | Verdadeiro | Falso | Verdadeiro | Verdadeiro -> Falso |
| Verdadeiro | Falso | Falso | Verdadeiro | Falso -> Verdadeiro |

AND / E, retorna verdadeiro se os dois valores retornam verdadeiros, caso um deles não seja verdadeiro, ele não retorna verdadeiro.

OR / OU, retorna verdadeiro se pelo menos um dos valores forem verdadeiros, caso ambos sejam falsos, ele retorna falso.

NOT / NÃO, inverte o valor, NÃO Verdadeiro, vai ser falso.

Para o SQL, a função AND, retornara todos os valores em que seus registros sejam positivos.

# SQLite Funcionalidades

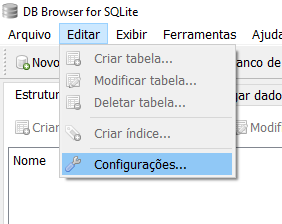
**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

Caso estiver usando o Windows, basta localizar o diretório de download do seu computador e abrir o executável DB Browser for SQLite. Para rodar no Ubuntu / Mac, basta usar o mesmo diretório ou o comando *sqlitebrowser* no terminal.

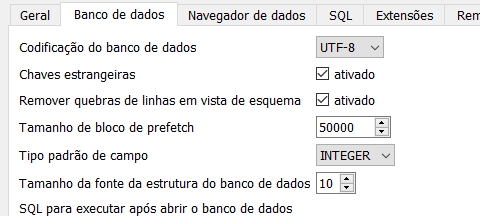
**

## Interface do SQLite

A interface do SQLite é bem simples. Antes de começar o desenvolvimento, vamos conhecer um pouquinho mais sobre essa ferramenta.



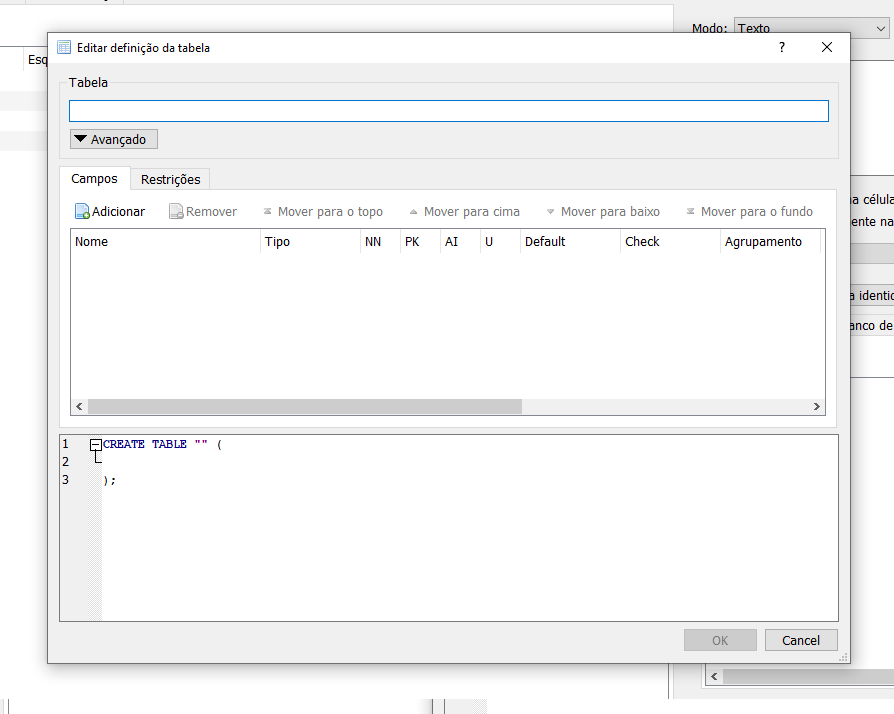
Editar, Configurações, para checar os primeiros requisitos, se estiver em outra linguagem, tem a tradução para português.



Na guia Banco de Dados, certifique-se que a configuração está UTF-8.

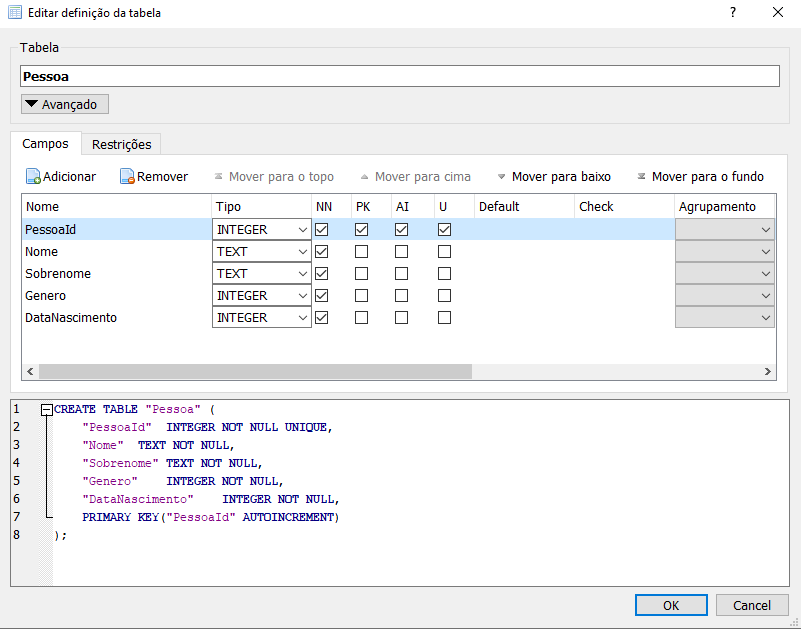
## Primeira Base de Dados

Para criar um novo banco de dados, basta clicar no botão ‘**Novo Banco de Dados**’ no canto superior esquerdo, irá subir um popup para você escolher onde quer salvar seu banco de Dados, recomendo salvar na Area de Trabalho com algum Nome.



Após selecionar a área onde deseja salvar seu banco de dados, novamente outro *popup* aparecera, nesse *popup*, são as configurações básicas de sua nova tabela, pois a base de dados você já criou.

Informações da nossa tabela:



No SQLite você consegue visualmente adicionar as colunas para começar criando sua tabela, basta clicar em Adicionar, repare que sempre começa com o **NomeDaTabelaId**, que no caso ali, foi a tabela Pessoa, sendo a primeira coluna, a PK.

Reparem que a *checkbox* **NN** significa, **NOT NULL**, ou seja, não pode possuir valor nulo, como foi citado anteriormente, a *checkbox* com a opção **PK**, é justamente a opção para identificar que aquela coluna é a chave primária, e opção **U**, é para que todos os valores sejam únicos ‘*Unique’*, reparem que no código, tudo que foi comentado, aparece ali também.

Futuramente iremos entrar em mais detalhes.

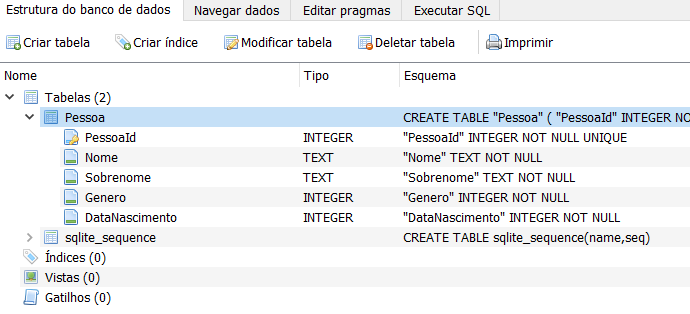
*Coluna Nome é do tipo* ***Text****, Not Null, pois não pode pessoas sem nome na nossa tabela.*

*Coluna Sobrenome, segue o mesmo estilo da coluna nome.*

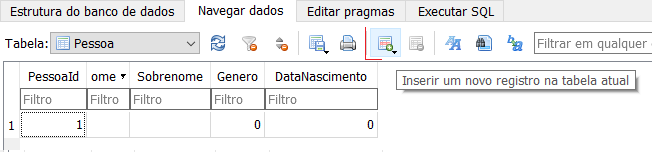
*Coluna Genero, podia ser* ***Text*** *também, mas estou usando* ***INT****, sendo 1 masculino e 0 feminino.*

*Coluna* ***DataNascimento****, como não tem a opção* ***date****, optei em deixar* ***Text****, esse estilo de escrita é chamado de* ***CamelCase****, ou seja, as primeiras letras maiúsculas e sem espaço, existe também a notação.*

Agora com a primeira tabela criada, ela não possui valor nenhum, podemos navegar para visualizar nossos registros clicando em ‘Navegar Dados’, mas não registramos nada ainda.

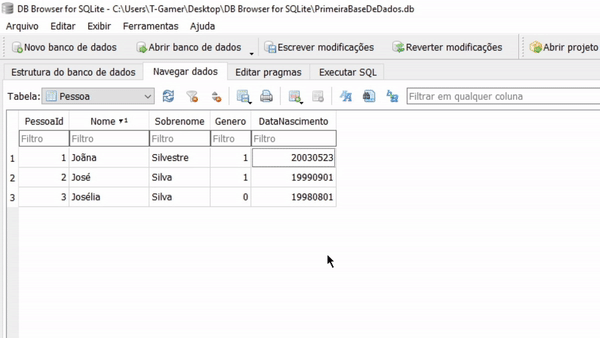


Para popular nossa base de dados, no SQLite, basta ir na aba denominada de Navegar Dados.



Reparem que, o Index e a PK já foram automaticamente preenchidos, devido as instruçoes que aplicamosna criação da tabela. Para preencher os valores, basta clicar na celula onde você quer preencher os valores.

Alguns valores tambem não nulos, o proprio SQLite tambem vão ser preenchidos.



Gif com as instruções.

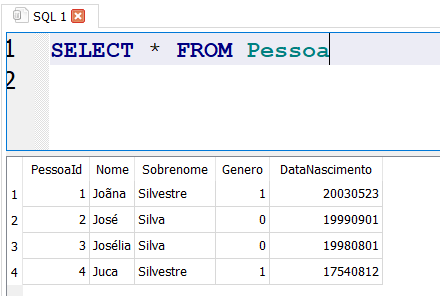
## Consultas Básicas

Agora que criamos a tabela e populemos os dados, está na hora de realizar as consultar com instruções, na lista de abas superiro, tem a aba Executar SQL, nessa aba, na primeira caixa, é onde iremos inserir as instruções.

Neste primeiro tópico, será abordado questoes simples como sintaxe e os primeiros comandos.

### Sintax e Select

Lembra que no primeiro capitulo, comentamos sobre os tipos de instruções, pois bem, vamos utilizar a mais básica de todas, uma instrução DML para selecionar todas as colunas e registros da minha tabela pessoa.

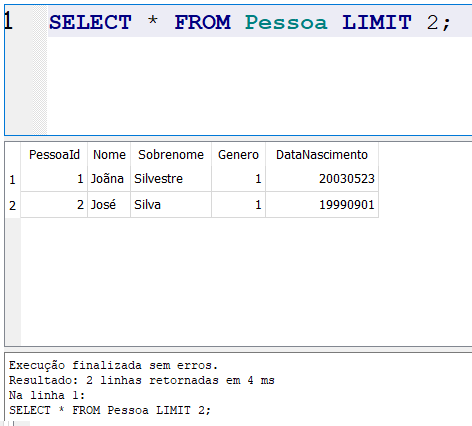


Para aumentar a letra das instruções, basta clicar no CTRL e usar o Scroll do Mouse. Em notebooks, basta apertar Fn +

* Apertanto TAB, o SQLite autocompleta.
* Para a instrução funcionar, basta apertar F5.
* Para comentar as instruções no SQL, ou seja, as instruções comentadas não serão executadas, basta usar /\* Comentário \*/ ou -- Comentário.

SELECIONE \* (Todos os Dados) Da minha tabela ‘Pessoa’.

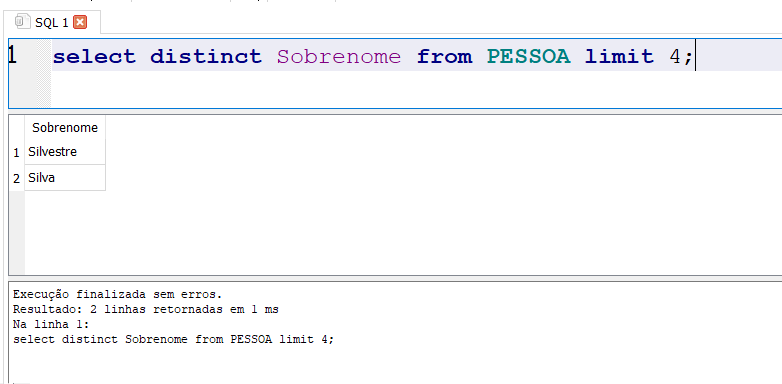
Esse é a instrução SQL mais fácil, porem é uma das mais perigosas, imagine-se em um cenário que você esta manipulando uma base de dados imensa, caso usar um comando desse sem especificação, poderá travar a consulta ou ate mesmo acabar com a memória do computador.

Para conseguirmos reverter esse problema, existe uma instrução que podemos limitar essa consulta para retornar um valor que pode ser determinado.

Dois pontos, é bem simples não, apenas acrescentar a palavra LIMIT e a quantidade de registros que eu quero visualizar.

O ponto e virgula ‘ ; ’ no final, é uma sintaxe do próprio SQL, e é opcional.

Vamos supor que na minha tabela, tenha vários valores iguais, para eu separar esses valores, e buscar apenas o valor único, uso a instrução para distinguir esses registros, parecido com o comando limite, esse de distinguir funciona da mesma forma, a palavra em inglês para distinto é ‘*Distinct*’, essa é a instrução no SQL.

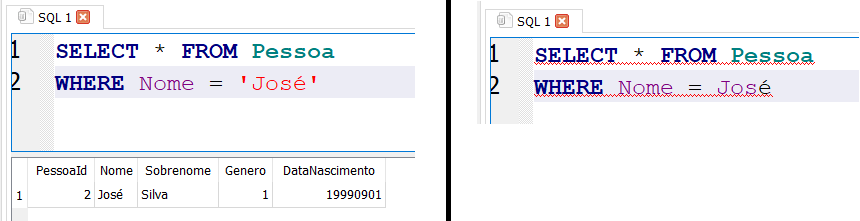


SQL não é case sensitive, ou seja, letras maiúsuclas e minusculas funcionam igual. Na minha instrução, estou selecionando todos os sobrenomes distintos da minha tabela pessoa, com um limite de 4 registros, porem, tenho somente dois registros unicos de sobrenome na minha tabela que serão retornados.

### Consultas com Where

A instrução **WHERE**, a tradução básica, seria **‘Onde’**, podemos fazer uma ligação com linguagens de programação, seria parecido com o *if*, que retorna determinados valores, *se* seu resultado for positivo.

O *Where* funciona da mesma forma, passamos uma **condição**, ele vai percorrer em todos os registros, quando encontrarvalores que correspondem a essa condição, vai retornar o determinado registro.

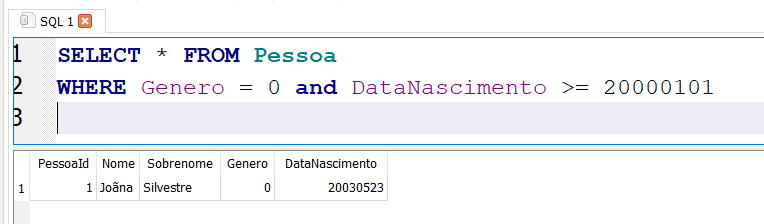


SELECIONE todos os registros DA tabela Pessoa

ONDE Nome seja igual a ‘José’.

Na segunda imagem, a instrução esta errada, o interpretador SQL até ficou com um erro, pois o valor José, não esta sendo representado com aspas, ainda especifiquemos durante a criação dessa tabela, que apenas aceitariamos registros do tipo string.

Podemos usar os operadores lógicos tambem durando as intruçoes WHERE.



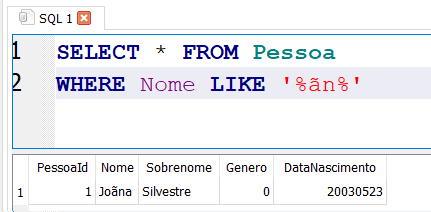
Selecione todos os registros da tabela pessoa

Onde o genero é igual a zero **E** a DataNascimento é maior ou igual a 20000101

Caso no lugar do operador **E**, estivesse o operador **OR**, minha consulta seria retornada com todos os registros que tenham o genero igual a zero tambem.

#### Palavra Reservada Like

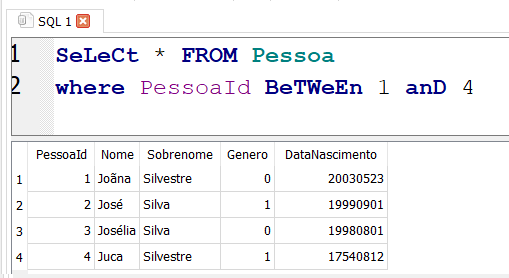
Mas, caso eu não saiba exatamente o valor de uma tabela, então e usado a instrução **LIKE**, essa instrução, é geralmente utilizada em textos, quando não sabemos exatamente oque tinha escrito no texto, apenas uma parte, passamos junto uma mascara, o ‘%’ que pode ser antes do texto ou depois, ou antes e depois, nesses casos, usar a porcentagem antes do texto com o Like, afirma que, estou procurando palavras que começam col algum valor, porem terminam com oque eu digitei, mesma coisa para os outros casos.



Selecione todos os registros da tebela Pessoa, onde seu nome **é parecido / como** ‘algum valor antes de **ãn** algum valor depois ’.

#### Palavra Reservada BETWEEN

Podemos localizar registros passando uma lista de registros, que no caso é a instrução **BETWEEN**, que significa ‘Entre’.

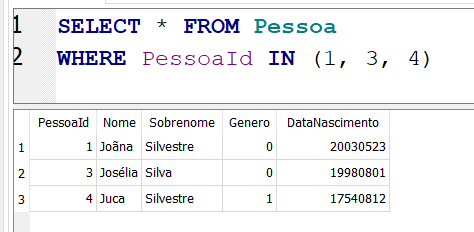


Reparem que, SQL, funciona com letras maiúsculas e minusculas juntos, mas não é muito agradavel de ver essas instruções dessa forma, apenas coloquei para você visualizar.

Selecione todos os registros da tabela pessoa, onde o id da pessoa esteja **EnTrE**, um e quatro.

#### Palavra Reservada IN

Com a istrução **IN**, ‘Em’, passamso realmente uma lista de valores para serem selecionados.



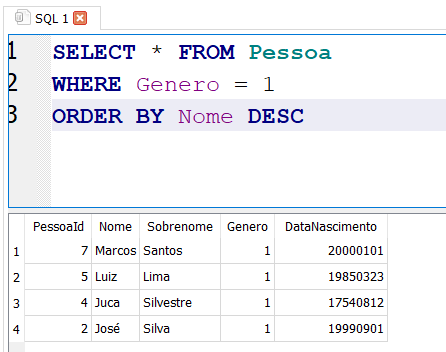
Selecione todas as pessoas da tabela pessoas, onde o id dapessoa esteja **em** (1, 3, 4).

### Trabalhando com Ordenação

Podemos ordenar nossas os resultados da nossas consultas com o comando **ORDER BY**, ‘ordenar por’, como tinha comentado antes, a sintaxe dessalinguagem é bem facil de entender.

Essa instrução para ordenar, devemos passar a coluna na qual vai ser ordenada e o sentido, por padrão é ASC, crescente, ou seja, começando do menor para o maior, mas podemos passar DESC, para decrescente.

Repare que a tabela agora possui alguns valores a mais, no proximo tópico, vamos adicionar mais registros a nossa tabela.

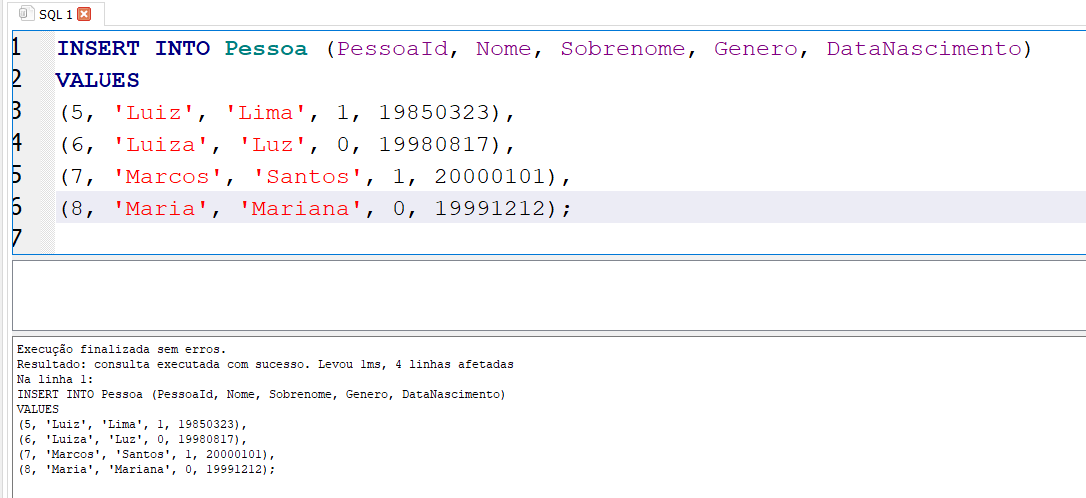


A instrução de ordenação tambem funciona tambem com o indice da coluna, ou seja, a PessoaId é o indice um, o Nome, essa coluna é o indice dois, e assim por diante, faça o teste alterando a palavra Nome, pelo valor dois, caso a colunana sua base de dados, cuja posição dois é a coluna com os registros de Nome.

### Instruções com Tabelas

#### Palavra Reservada INSERT INTO

Para adicionar mais valores a tabela, usamos a instrução INSERT INTO, ‘insira em’.

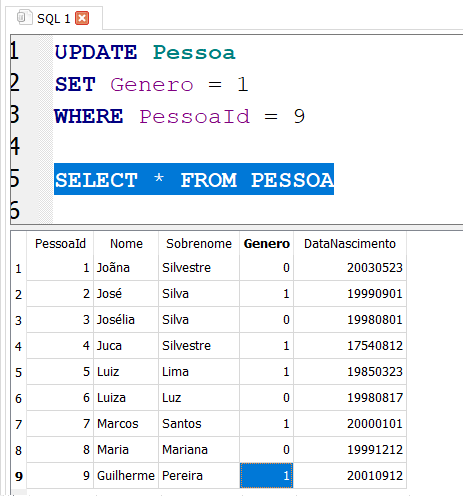


Insira na tabela Pessoas (PessoaId, Nome, Sobrenome, Genero, DataNascimento), Passamos entre parenteses, as colunas da nossa tabela.

Valores (Id, Nome, Sobrenome, Genero, DataNascimento), Passamos entre parenteses, os valores correspondentes aos valores da tabela, separados por virgula eo ultimo valor com ponto e virgula.

#### Palavra Reservada UPDATE e SET

Caso algum valor da minha tabela esteja errado, é possivel atualizar somente aquele valor com instruçoes proprias da linguagem SQL. para isso, usamos a instrução **UPDATE**, adicionei mais um valor a minha tabela com o genero feminono, porem seu nome é ‘Guilherme’, claramente, esse é um erro input, que o propio usuario pode ter selecionado errado ou o banco de dados armazenou errado



Reparem que para executar apenas determinadas instruções em SQL, basta selecionar a instrução e apertar F5. Usamos duas palavras reservadas, a palavra **UPDATE** e a palavra SET. para atualizar a tabela pessoa, para o gênero igual a um, onde o id da pessoa é igual a 9, poderíamos utilizar outro valor para representar no lugar do Id, mas o Id é confiável e único, podia estar manipulando outro valor, caso, o nome Guilherme estivesse sido registrado mais de uma vez, as instruções seriam aplicadas a ambos os registros.

Caso eu não especifique o **WHERE**, a instrução de update será aplicada a todos os registros da minha tabela.

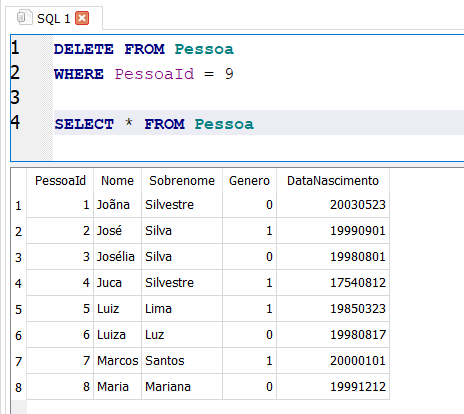
* Para executar apenas determinada instrução em SQL, basta selecionar o conjunto de instruções e apertar F5.

#### Palavra Reservada DELETE

A instrução delete, como o nome já sugere, serve para deletar os registros da minha tabela, parecido com a função updade, requer uma condição com o **WHERE** para não deletar todos os registros da minha base de dados.

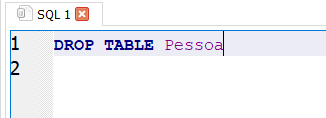
A instrução é bem simples, vou deletar o valor que tínhamos atualizado com a função update, o ‘Guilherme’.

* O SQLite já faz o *commit*, ou seja, não precisamos especificar o *commit* da consulta, o *commit* nada mais é que uma instrução que envia nossas alterações.



#### Palavra Reservada DROP

Ultimo Conceito básico de SQL, a instrução **DROP TABLE**, como todos as instruções de SQL, servem para deletar uma tabela, diferente da *DELETE*, essa aqui deleta todos os valores.



Sim, esse é o comando, após executar, repare que a Tabela pessoa até mudou de cor, pois ela não existe mais em sua base de dados.

Agradeço por chegar até aqui, agora tem uma listinha básica de 5 exercícios para prosseguirmos ao capítulo seguinte.

As atividades estão no final do capitulo, é o penúltimo link nas referências bibliográficas.

# SQL Server Funcionalidades

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

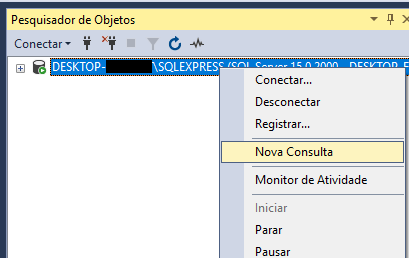
Antes de começar os conceitos mais avançadinhos em SQL, recomendo fazer o download do SQL Server, para aprender mais uma ferramenta.

Para realizar o download dessa ferramenta, esta em um segundo documento na lista de documentos.

Após ter realizado o download o SQL Server e de seu gerenciador de banco de dados, basta abrir o seu Microsoft SQL Server Management Studio 18, digitando *ssms*, na barra de pesquisa do Windows (Caso estiver usando o Windows) e clicar em conectar.

## Fundamentos das Tabelas

Apesar de já termos criado uma Tabela no capitulo passado, vamos criar mais uma, porém, antes, temos que criar um novo banco de dados no SQL Server.

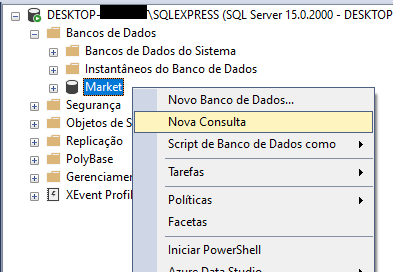


Para isso, vamos criar tudo com instruções SQL, para começar, clique com o botão direito na sua base de dados do seu computador local e, em nova consulta.



Vou utilizar nomes do banco de dados e das tabelas em inglês, mas você pode escrever os nomes em português.

* Para deletar uma base de dados é *DROP DATABASE Nome.*



Seu banco de dados vai estar armazenado na pasta como mesmo nome, agora vamos acessar para criar nossas tabelas.

### Palavras Reservadas para Tabelas

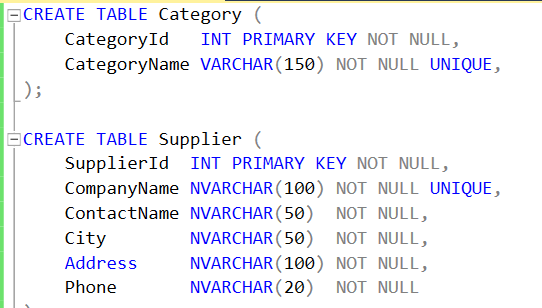


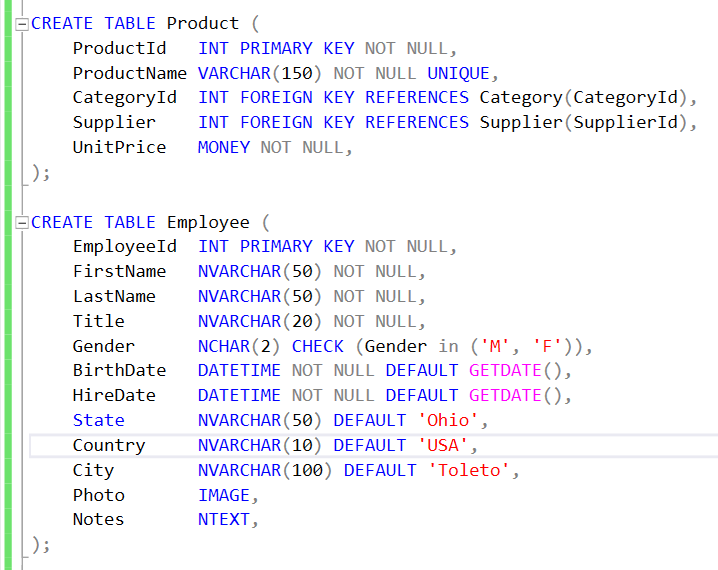
Tabela Categoria, estou passando dois parâmetros.

*CategoryId* é um valor inteiro chave primária não nulo.

*CategoryName* é uma coluna do tipo *varchar* até 150 caracteres, não nulo e **Único.**

A palavra reservada *Unique*. Aplicada a coluna, essa coluna não pode ter registros duplicados, ou seja, no nosso exemplo de mercado, não queremos categorias de produtos que sejam iguais.

A tabela *Supplier*, é fornecedor em inglês, essa tabela possui mais colunas que a tabela anterior, possui uma chave primaria, e o único diferencial é o *CompanyName,* que é único.



A tabela *Product*, que inglês significa produto, tem os mesmos requisitos da tabela anterior, contudo, a *CategoryId, Supplier,* são referências de outras tabelas, ou seja, são as chaves estrangeiras e a *UnitPrice*, preço unitário, seu formato é de uma palavra reservada para representar o dinheiro.

Tem inclusive um erro de digitação, pois faltou o Id, esse erro vamos corrigir ao decorrer do guia com instruções do próprio SQL.

Sobre a tabela funcionários, essa tabela contem varias informações que já foram citadas, porem a coluna *Gender****,*** está com mais uma palavra reservada, a palavra **check**, é uma *constraint*, uma limitação que podemos impor para que, nessa coluna só seja aceito os registros que sejam entre ‘M’ e ‘F’. reparem que eu guardo esses registros em parênteses.

Na coluna *BirthDate* e *HireDate,* seus registros possuem três palavras reservadas novas, a instrução **Datetime, Default** e **Getdate()**, sobre o datetime já foi citado e explicado nos fundamentos de SQL.

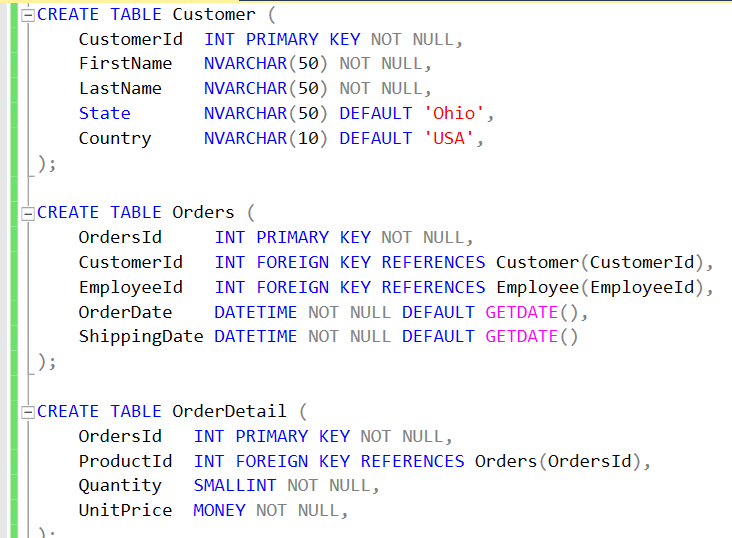
A palavra reservada **Default**, caso nenhum valor seja registrado nas colunas com essa instrução, os valores vão ser aplicados a *função* ao lado, que é a *Getdate(),* essa função pega a data atual durante o registro.

Essa mesma logica do default foi selecionado e aplicado nas próximas colunas, para obter valores default.

A coluna foto, o valor de seu registro é binário, porem no SQL Server, possui um nome reservado especial, o **Image**. Já a coluna ‘Note’, é um tipo de texto normal.

Essa próxima coluna, é na verdade um desafio especial.

O seu desafio é fazer uma listinha da tabela, comentando a respeito das suas colunas, por exemplo, ‘Na tabela *Employee*, a primeira coluna é a *EmployeeId*, essa coluna significa que é a chave primária, a coluna que representa essa tabela, o tipo de dado aceito por essa coluna é do tipo inteiro e essa coluna não aceita valores nulos. Aliás, você sabia que essa *constraint* possui embutido as funcionalidades de *Unique* e *Not Null*, sendo assim eu não preciso especificar fora’.

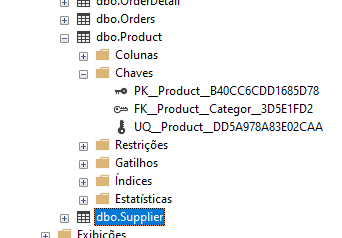
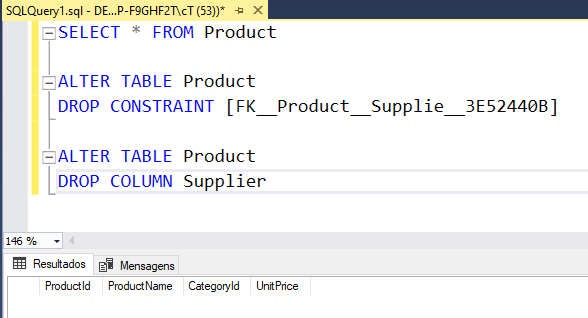


### Alterar e modificar colunas

#### Deletar determinadas Colunas

Essa instrução é muito poderosa, ela que altera a coluna, deleta a coluna e modifica a mesma, essa é uma das instruções que não pode sair da sua caixinha de instruções da linguagem SQL.

Como tinha citado, ela possui varias funcionalidades, a primeira e mais simples, é a instrução para deletar uma tabela, vamos usar as palavras reservadas, **Alter**, **Table,** **Drop** e **Constraint**.

****

Para remover uma chave estrangeira no SQL Server, antes é preciso desanexar a chave da tabela, para isso é usado o *drop constraint* e a localização da chave.

* Você pode arrastar a chave e colocar ela na parte da sua instrução.

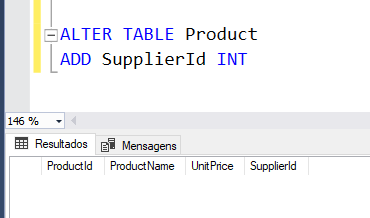
Depois de desanexar a chave, vamos dropar a tabela com a instrução *drop column* e passando a tabela em especifico para realizar a deleção.

Provavelmente vai dar um erro, mas é só atualizar a base de dados e as colunas já vão ter sido deletadas corretamente.

Geralmente realizar instruções com o SQL Server em chaves primárias e chaves estrangeiras é, em algumas vezes complicada para iniciantes.

#### Adicionar novas Colunas

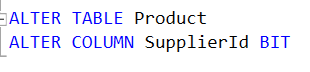
Para adicionar coluna no SQL é bem simples, basta usar o mesmo comando abordado no tópico acima, alterando a instrução de **DROP** para **ADD**, porem, temos que especificar o tipo de dados que essa coluna vai aceitar, no exemplo, usei o tipo inteiro.



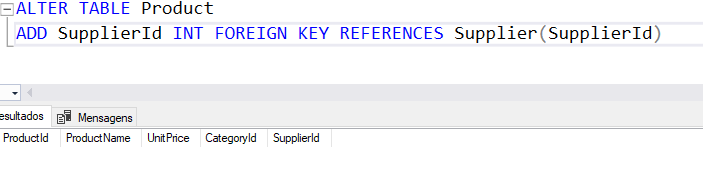
Sem muitos comentários, essa instrução é bem fácil de entender e aplicar.

#### Modificar tipo de dado da Coluna

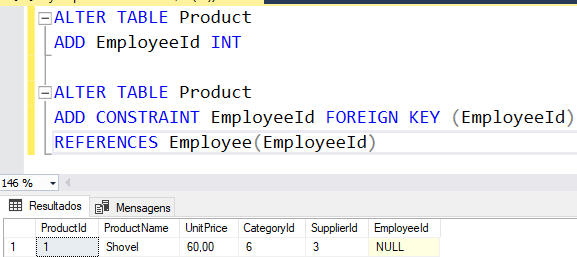
Essa instrução, como a anterior, é bem simples, vamos passar a coluna da nossa tabela e modificar o seu atual tipo de dado, as vezes não é possível alterar devido a já ter dados na coluna, quando acontece isso, usamos outras técnicas.



Para modificar e colocar para chave estrangeira a coluna *SupplierId*, uma das formas é criar ela já com as configurações de uma chave estrangeira.



A outra forma é fazer uma referencia já em uma coluna de uma tabela criada, ou seja, alterar o tipo da coluna para o tipo de chave estrangeira.



Alterar tabela Produto,

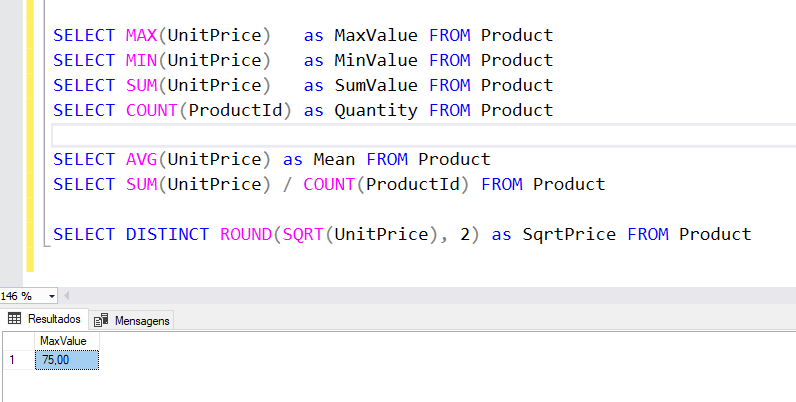
Adicionar *constraint* a coluna *EmployeeId* do tipo chave estrangeira (*EmployeeId*)

Referência a tabela *Employee* (*EmployeeId*).

Caso já tiver registros, a coluna vai assumir o valor nulo.

### Funções e Apelidos

Como já foi citado anteriormente, podemos fazer inúmeras contas matemáticas, mas existe algumas funções que a própria linguagem SQL que nos oferece.



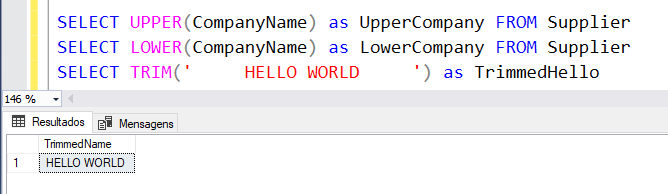
As funções são palavras reservadas, que recebem como parâmetro as colunas que você quer realizar as instruções.

A função **MAX**, retorna o maior valor registrado na coluna que foi indicada entre parênteses, a função **MIN**, retorna o menor valor registrado, a função **SUM** soma todos os valores registrados na coluna indicada, e a função **COUNT**, conta a quantidade de valores registrados.

A função **AVG**, retorna a média de valores da coluna especificada, e também podemos aplicar os operadores matemáticos nos resultados das funções, ou seja, o resultado da instrução após a instrução da média, é a própria média escrita de outra forma, a soma de valores dividido pela quantidade de valores, ambos as instruções retornaram à média.

Podemos criar algumas funções aninhadas, funções dentro de outras funções, a ultima instrução, vai retornar os valores *únicos*, onde estou aplicando a função **SQRT**, que retorna a raiz quadrada, porem o resultado dessa função esta em outra função, a **ROUND**, essa instrução, reduz os caracteres depois da virgula, como muitos valores da raiz quadrada vem com virgula, foi aplicado uma função de arredondamento, passando o valor para ser arredondado e a quantidade de casas após a virgula, que no exemplo da imagem, foram duas casas.

Reparem que após a função, existe outra palavra reservada, a palavra **AS**, essa instrução indica que a instrução após, vai ser um apelido, podemos entender isso como o nome da coluna, pois, após aplicar alguma função, a coluna não vai possuir nome, assim podemos dar um nome a essa nova coluna criada, reparem que nos resultados, a coluna está com o nome de MaxValue. Futuramente usaremos muitas essas funções e os apelidos.



Similarmente, temos funções para valores no formato *string*, existem muitas outras funções, mas esse é um guia básico.

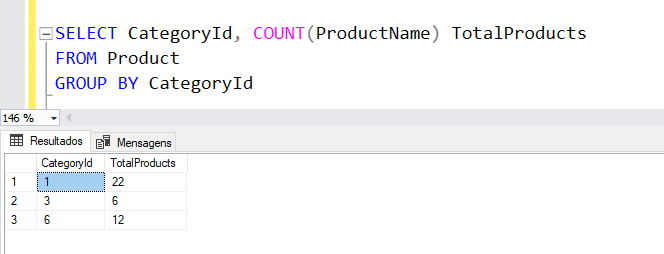
A função **UPPER**, converte todos os registros passados como parâmetros para maiúsculo.

A função **LOWER**, converte todos os registros para minúsculo.

A função **TRIM**, é a mesma da estatística, sua tradução é ‘aparar’, o registro passado como parâmetro, caso o usuário tiver digitado algum espaço a mais na digitação de seu formulário, a função vai remover esse espaço.

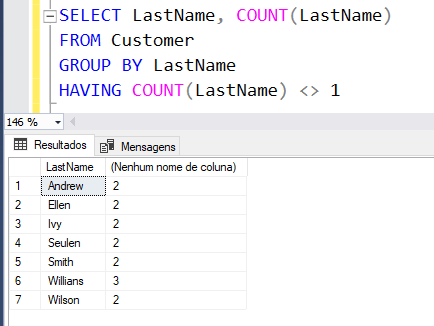
### Instruções de Agrupamento

Uma das formas de agrupar valores no SQL é com a instrução **GROUP BY**, essa instrução m fácil de compreender, basicamente, vamos agrupar valores aplicando alguma dessas funções citadas a cima a alguma coluna com registros em alguma tabela da nossa base de dados.



Nessa instrução, está sendo agrupado com a função **COUNT**.

Selecionar a *CategoryId*, contar a quantidade de produtos e apelidar de ‘*TotalProducts*’ da tabela *Products,*e agrupar pelas *CategoryId*.

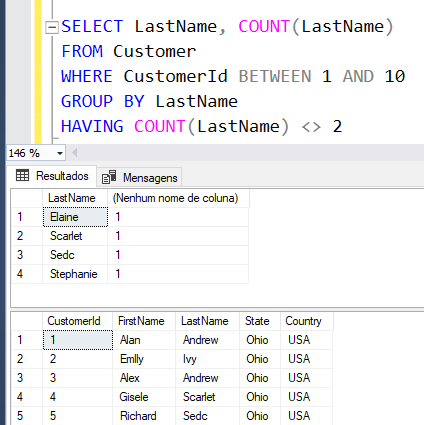


É possível aplicar uma instrução parecida com o *Where*, que é especifica para agrupamentos, a instrução **HAVING**, essa instrução só funciona após o agrupamento, juntamente com uma função.

Estou selecionando, todos os clientes, que possuem o Sobrenome diferente do registro com valor um.

Outro ponto legal para compartilhar nesse guia, é que na minha instrução, não apliquei o apelido, sendo assim, essa nova coluna gerada a partir da minha consulta, não possui nome.

Caso queiram dicas para popular essa base de dados, no penúltimo capitulo desse guia, tem a lista de referências, e lá tem um dos sites que obtive algumas referencias para colocar na base de dados.

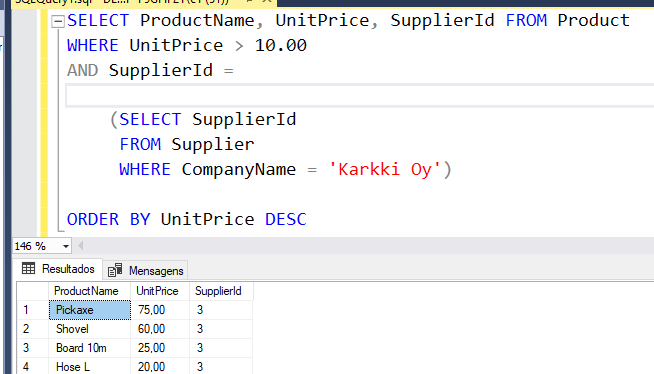
Podemos sem problema utilizar instrução *Where*, para realizar consultar mais sigilosas.

Nesse caso, ele fica antes go *Group by*.

O resultado abaixo, é parte da seleção total.

### Sub Consulta

A Sub consulta ou consulta ‘aninhada’, ou em inglês, **SubQuery**, é uma consulta dentro de uma instrução **WHERE**, nessa sub consulta, as colunas dentro da sub consulta, devem corresponder as consultas externas e não pode conter ordenamento em uma sub consulta.



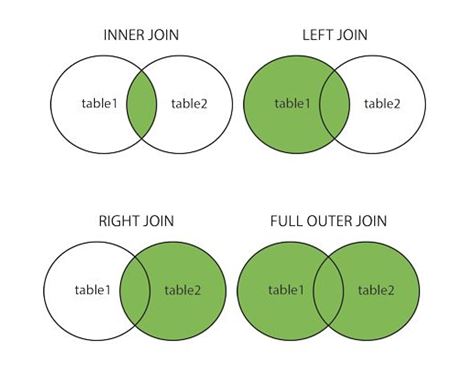
A consulta na imagem, estou utilizando duas tabelas diferentes para obter os registros, usei um espaço chamado de indentação, apenas para ficar mais fácil de visualizar.

A instrução *Where* é a responsável pela instrução da sub consulta, estou buscando da tabela *Supplier*, todos os *SupplierId*, cuja *CompanyName* é igual a ‘*Karkki Oy*’, esse resultado, vai ser voltado ao meu *Where*, para concluir a instrução, retornando os devidos registros da minha consulta.

* Uma sub consulta também suporta outra sub consulta.

### Junções

As junções, ou em inglês **Joins**, a junção mais simples que existe é o **SELF JOIN**. Pois, apenas trabalhemos com mais de uma tabela na sub consulta, então, vamos agora utilizar, mais de uma tabela para realizar consultas ainda melhores.

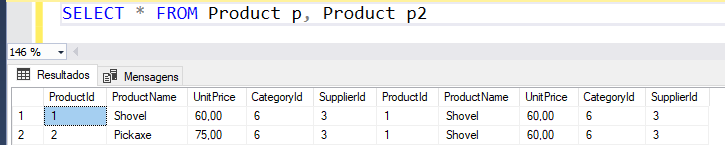


Repare na imagem ao lado, é uma imagem já bem utilizada, porém, representa bem o que as junções fazem, são divididos em cinco categorias, que iremos, ao decorrer desse tópico, comentar e utilizar esses tipos de instruções.

A junção mais fácil de compreender, é a Self Join, que não está desenhada da imagem.

#### Self Join

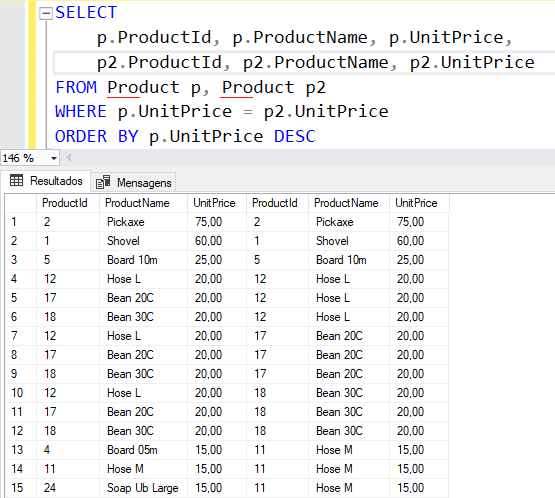
Onde a tabela se junta a ela mesmo. Sim, é isso mesmo, nessa consulta, entra alguns conceitos de correlação, chamamos a mesma tabela duas vezes, porem com apelidos diferentes.



A priori é meio confuso, mas entenda agora que, o nosso sgbd, fez todas as possíveis combinações da nossa tabela, para ela mesmo, meus simples quarenta linhas de registros, se tornaram mil e seiscentos registros.

Podemos aplicar filtros também, assim organizando melhor nossa consulta de correlação entre os registros, que no caso da nossa tabela são os produtos.

Nessa próxima consulta, vamos colocar em prática, selecionando apenas quatro colunas das duas tabelas, e comparando o valor de preço iguais para diferentes produtos registrados em abas as tabelas da minha base de dados.



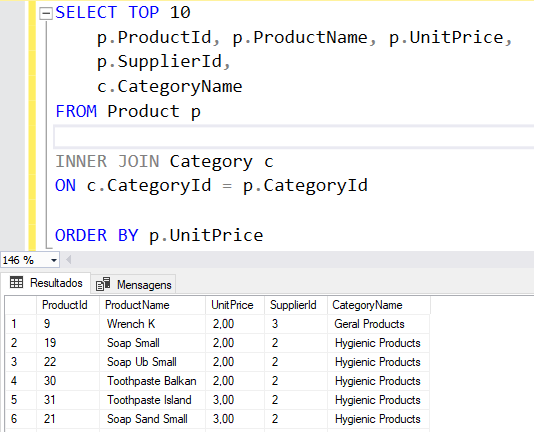
#### Inner Join

Diferente do *Self Join*, o **Inner Join**, ele irá retornar valores, que estejam, de alguma forma ligados pelas chaves, no caso, a *foreign key*, ou chave estrangeira, que é a chave primária de uma tabela que esta rendo referenciada por outra tabela.

Lembra da tabela *Product*, ela possui duas chaves estrangeiras.

A chave para a categoria em qual ela pertence e para o seu devido fornecedor, porém, para alterar o id e colocar o novo respectivo, usamos o **Join**.

Esse tipo de *join*, vai retornar valores que estejam tanto na tabela da *esquerda*, quanto na Tabela referenciada. Entraremos mais em detalhes sobre as tabelas no próximo join.

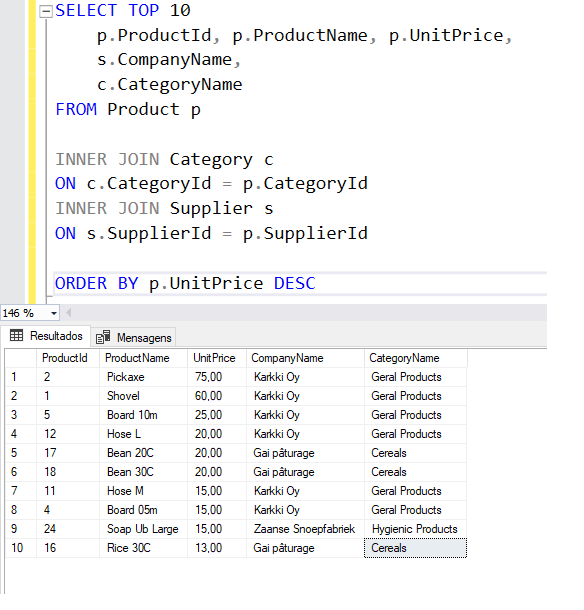


Similarmente ao *selft join*, usamos as palavras para apelidar as tabelas, após selecionar a tabela principal, vamos usar a palavra reservada **INNER JOIN**, e digitar a tabela a qual vai ser referenciada.

Usamos também outra palavra reservada, a **ON**.

Que, vai fazer a junção das tabelas, partindo das chaves que estão presentes nas duas tabelas.

Contudo, a tabela *SupplierId*, ainda está sem a devida identificação, podemos usar a mesma junção, e a instrução vai ficar assim.

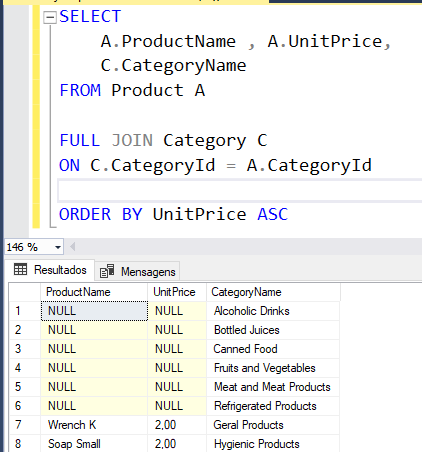


O Inner Join, ficou logo a baixo do outro Inner Join, sempre, sendo referenciado pela tabela mãe, que no caso é a tabela *Product*.

A instrução é a mesma que a anterior, só que, dessa vez, deixando a nossa consulta mais bonita e ordenando em forma decrescente.

#### Full Join

Esse *join*, vai juntar todos os registros da tabela principal e da tabela referenciada, ou seja, a tabela principal é a tabela da esquerda, e a referenciada é a tabela da direita.



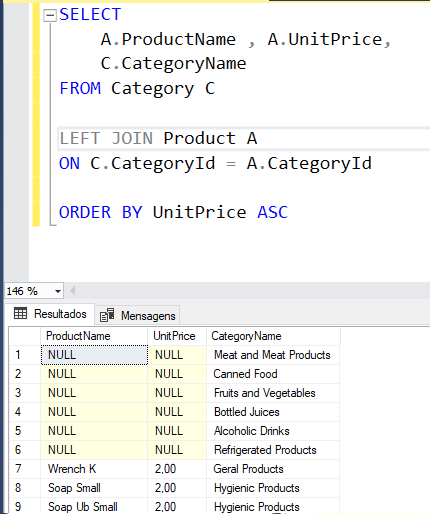
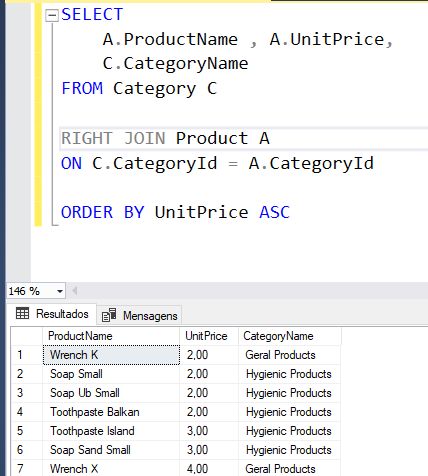
Esse *join*, retornou todos os registros, independentemente se já foram atribuídos a algum outro registro, ou seja, da minha tabela *Category*, existem registros, ou seja, algumas categorias que não foram associadas a nenhum produto.

Seguindo essa lógica, não existe nenhum valor atribuído a essa categoria, então esse *Join*, atribuiu todos os valores dessa consulta para nulos, lembrando que nas nossas tabelas não aceitamos valores nulos.

Porem, o *full outer join*, ou *full join*, faz essa associação quando, nenhum registro foi atribuído a sua referência, lembrando, eu poderia fazer a junção pela categoria referenciando a tabela *product*.

#### Left e Rigth Join

Quando juntamos as tabelas, temos dois lados, esquerda e direita, o lado esquerdo, é a tabela que você usa na instrução *from*, e a da direita, é a referenciada pela *roreign key.*



Inverti as instruções, consultando agora pela categoria, juntando pelo produto, reparem que, na primeira imagem, usei a coluna da direita, que são os produtos, assim, essa consulta vai retornar os registros que estejam em alguma categoria registrada, resumindo, nesse contexto, teve o mesmo resultado de uma consulta com instruções de *inner join*.

Já a segunda imagem, foi agrupado as categorias com os produtos, porém, os produtos sem registros também foram retornados.

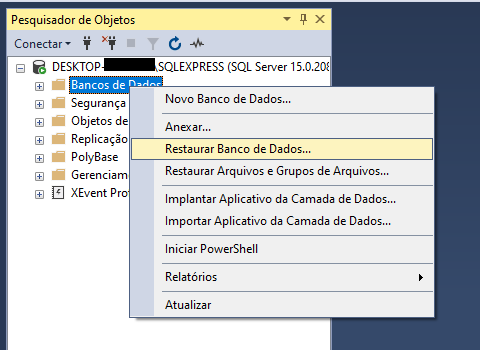
## Restaurando uma base de dados

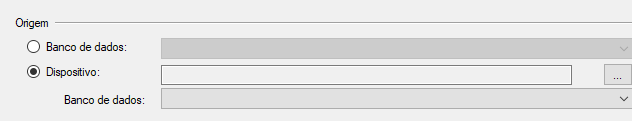
Já acabou a lista de atividades da pasta SQL Server Fundamentos?

Pois bem, vou mostrar nesse último tópico, como fazer uma restauração de uma base dedados real, basta fazer o download do arquivo .BAK.

<https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks/AdventureWorksLT2019.bak>

Para realizar o download, basta copiar o link.

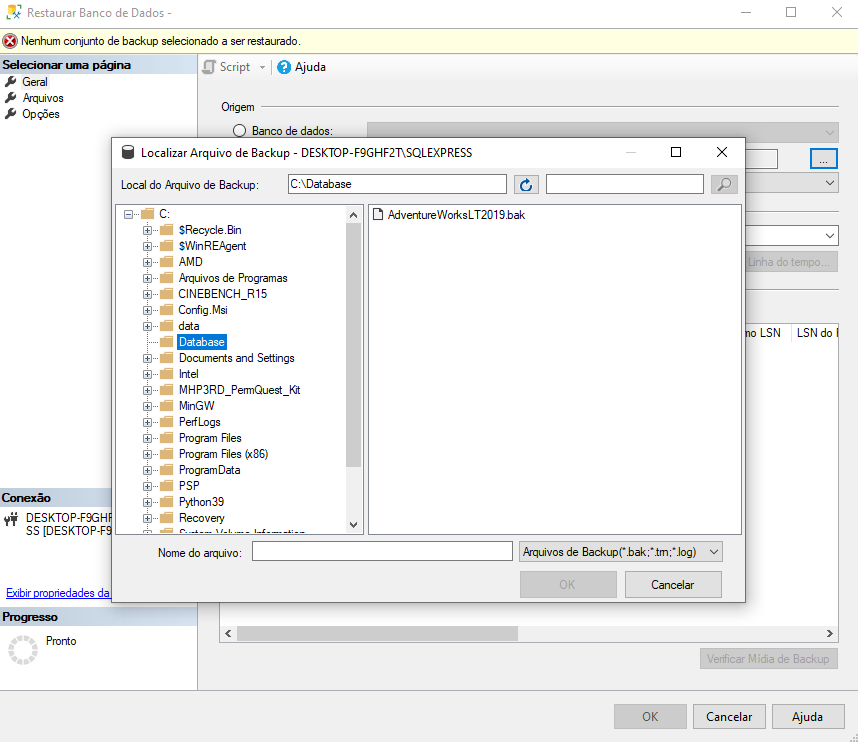




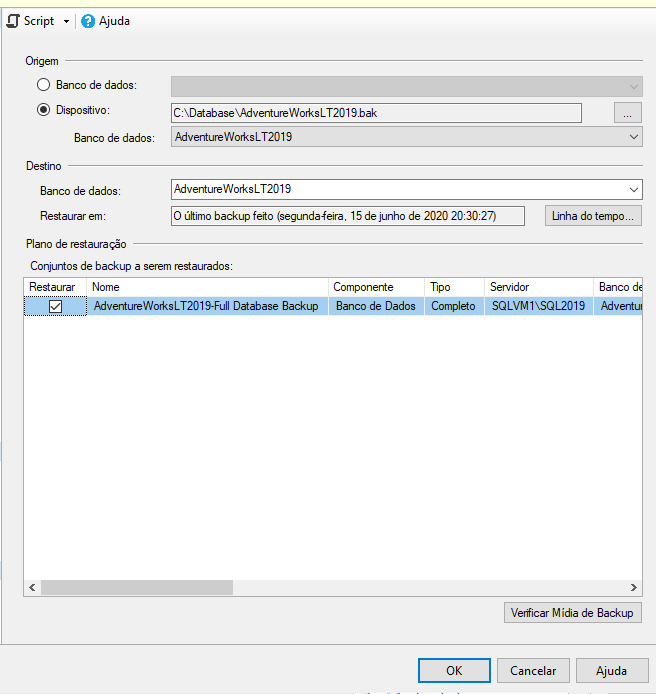
Clique em Dispositivo, após abrir o popup de informações da restauração da base de dados.

Observação, eu coloquei o download do arquivo *BAK*, em uma pasta no diretório C chamada Database.

Depois que achar o devido diretório, basta clicar no arquivo a direita e licar no botão adicionar.



C -> Database AdventureWorks.



Depois basta marcar a checkbox Restaurar e clicar em Ok.

Agora você tem uma base de dados novinha em folha para poder realizar suas consultas.

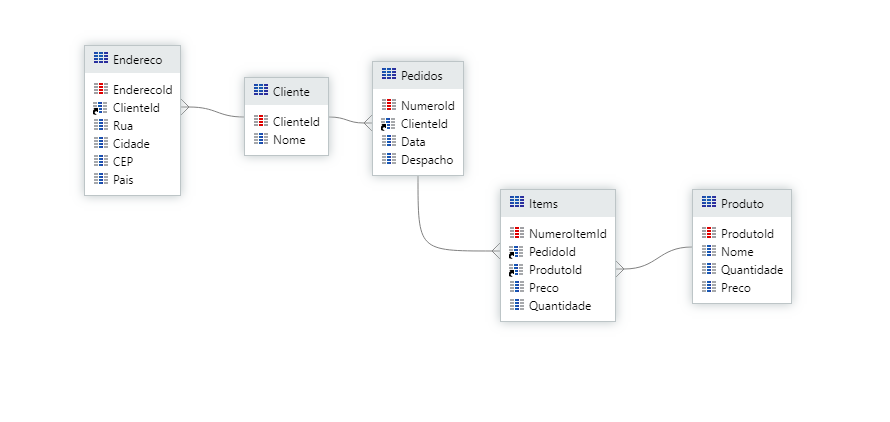
# Glossário

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

**Atributos**: Lembra da função filtro do Excel, onde você aplicava essa função nas colunas, pois bem, os atributos são as colunas.

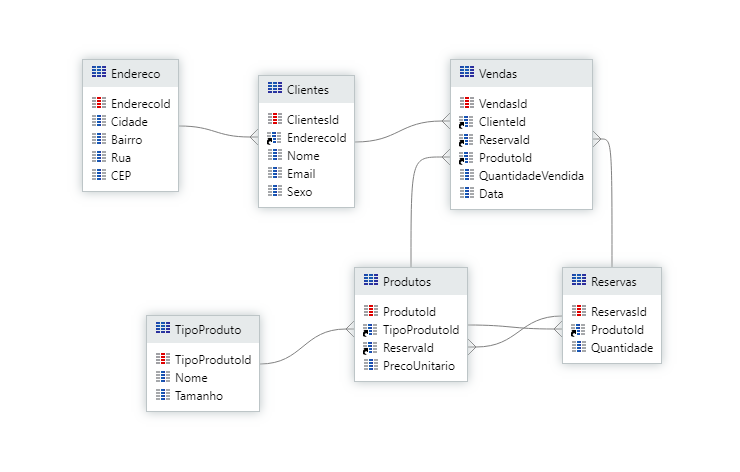
**Banco de Dados Relacional**: Imagine uma tabela no Excel, um banco de dados é isso. O conceito de banco de dados relacional, está, devido a relação dessas tabelas por chaves, veremos mais à frente esse conceito, mas, por agora, entendemos que são tabelas organizadas em linhas e colunas.

**Chave Estrangeira (FK)**: Quando uma PK está em na linha de outra tabela.



**Chave Primária (PK)**: Chave principal, usada para identificar e diferenciar registros, busca a unicidade das linhas.

* É possível somente uma chave primária por tabela.
* Uma tabela pode conter a chave primária de outra tabela.



**Checkbox**: São caixas selecionáveis, geralmente vemos essas caixinhas quando estamos respondendo algum formulário online ou validando as clássicas caixas de ‘Eu não sou um Robô’.

**Domínio**: Conjunto de valores que uma coluna pode possuir, se eu tiver valores numéricos, só poderei ter valores numéricos.

**Índice**: É uma lista de valores ordenados que apontam a posição das linhas, muito útil para agilizar a consulta.

**Integridade Referencial**: Conceito de relação das minhas tabelas, ou seja, a partir dos dados, eu não consigo fazer determinada ação devido a essa relação, exemplo, se houver registros relacionados eu não consigo remover esse determinado registo, como uma escada, não consigo subir estando já em cima.

**Normalização**: Redução das tabelas em tabelas menores como na última foto apresentada.

**RDBMS / SGBD** (Relational Database Management System): É uma GUI, um software que manipula nosso Banco de Dados Relacional.

**Registro**: São as linhas das tabelas, também chamada de Tupla, que podem conter valores nulos.

**Tabela**: Simples estruturas de linhas e colunas que contém uma Pk.

**Variável**: Em programação de computadores, a variável é um espaço na memória para armazenar determinado tipo de dado.

# Bibliografia

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

Documentação SQL (Microsoft) -> <https://docs.microsoft.com/pt-br/sql/?view=sql-server-ver15>

Visão Geral SQL (Microsoft) -> <https://docs.microsoft.com/pt-br/sql/relational-databases/databases/databases?view=sql-server-ver15>

Tipos de Dados (W3School) -> <https://www.w3schools.com/sql/sql_datatypes.asp>

Cadeia de Caracteres (Wikipedia) -> <https://pt.wikipedia.org/wiki/Cadeia_de_caracteres>

Guia de Referência (W3Schoolss) -> <https://www.w3schools.com/sql/sql_quickref.asp>

Palavras Reservadas (W3Schools) -> <https://www.w3schools.com/sql/sql_ref_keywords.asp>

Funções (W3School) -> <https://www.w3schools.com/sql/sql_ref_sqlserver.asp>

Base de Dados Ideia -> <https://www.researchgate.net/figure/Product-categories-and-the-number-of-products-in-each-category_tbl1_24068280>

Nomes em Inglês -> <https://www.ssa.gov/oact/babynames/decades/century.html>

Base de Dados da Microsoft -> <https://docs.microsoft.com/pt-br/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms>

Lista de Atividades SQLite -> <https://github.com/xGabrielR/SQL-Fundaments/tree/main/Atividades/SQL%20%26%20SQLite%20Fundamentos>

Lista de Atividades SQL Server -> <https://github.com/xGabrielR/SQL-Fundaments/tree/main/Atividades/SQL%20Server%20Fundamentos>

# Informações Adicionais

**━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━**

Lista de Atualizações:

Gabriel Richter. - *30/07/2021*

Gabriel Richter. - *09/08/2021*

Gabriel Richter. - *11/08/2021*

Gabriel Richter. - *12/08/2021*