

SQL Guide

with practical exercises



Gabriel Richter

Sumário

Motivo de Aprender SQL	5
Introdução ao Pensamento Analítico	5
Como Manipular esses Dados.....	6
Significado da sigla SQL.....	6
Qual Ferramenta Usar?.....	7
SQL Fundamentos dos Dados	9
Tipos de dados Literais e Caracteres.....	9
Tipos de dados Numéricos.....	10
Tipos de dados Temporais.....	11
Tipos de dados Booleanos.....	12
Palavras Reservadas.....	12
Conceito de Chaves	13
Operadores Relacionais e Matemáticos	14
Operadores Lógicos	14
SQLite Funcionalidades.....	16
Interface do SQLite.....	16
Primeira Base de Dados	17
Consultas Básicas.....	20
Sintax e Select.....	20
Consultas com Where.....	23
Trabalhando com Ordenação.....	26
Instruções com Tabelas.....	27
SQL Server Funcionalidades	30
Fundamentos das Tabelas.....	31
Palavras Reservadas para Tabelas.....	32
Alterar e modificar colunas	34
Funções e Apelidos.....	37
Instruções de Agrupamento	39
Sub Consulta.....	40
Junções	41
Restaurando uma base de dados.....	47
Glossário.....	51
Bibliografia.....	53
Informações Adicionais.....	54

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: O capítulo um, nesse capítulo será citado um pouco sobre negócio e SQL, como ela funciona e download do SQLite. Recomendo acessar o glossário no final desse guia 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.

Abstract

This mini guide is divided into three parts.

1. Introduction: Chapter one, in this chapter will be mentioned a little about business and SQL, how it works and download SQLite. I recommend accessing the glossary at the end of this guide to understand a little more of the terms will be using in this guide. Maybe in the future I will complement it with NoSQL like MongoDB.

2. First Commands: In chapter two, will be like first queries and creating a first Database all with a free, soft and mobile software, the SQLite.

3. Professional Software's: Second data base software used in this guide is SQL Server, a complete *dbsm* for more complex queries.



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.

Introdução ao Pensamento Analítico

Realizar uma consulta em uma base de dados é fácil, você vai aprender nesse guia, mas você deve entender o porquê você está fazendo essa consulta, você está ajudando um cientista de dados em um projeto que requer tais manipulações, você apenas quer analisar os dados pois um superior pediu um resultado, ou apenas treinar SQL.

A maioria das ferramentas no mercado nasceu com o intuito de resolver um problema que o indivíduo estava passando no momento em que foi proposto a ideia dessa ferramenta, se ainda não estiver convencido, olhe o exemplo do *visual studio code*, um ambiente de desenvolvimento integrado muito completo por sinal.

Fazendo uma pequena pesquisa pela internet, vemos muito ênfase na frase “Loja de extensões imensa”, entre outras frases, ou seja, essa ferramenta, possui muitas extensões onde por meio delas é possível manipular *scripts* de diversas formas, não vou entrar mais em detalhes pois acho que você já entendeu.

Recomendo que ao decorrer das atividades desse documento, você imagine-se em um ambiente de trabalho, e após realizar as determinadas atividades, faça um relatório, ou entenda mais por que você fez aquele exercício.

Fazendo esses exercícios extra de pensar no negócio, garante uma preparação antes mesmo de desenvolver outros projetos sem ser especificamente com SQL, ou seja, você recebeu um problema, tente interpretar esse problema que você recebeu e quebre em tarefas menores, “Quero que você faça uma base de dados para frutas”, você realmente entendeu essa pergunta?

Pergunte a unidade de negócio que lhe fez essa pergunta o motivo, entender a causa raiz ajuda a encontrar potenciais métodos e formas para conseguir concluir esse desafio, pois você pode desenvolver uma solução que na maioria das vezes não vai servir ao propósito, pois existem várias formas de desenvolver uma base de dados para frutas, mas que tipos de frutas, ou o que deve conter nas tabelas dessa base, inclusive, essa unidade que lhe pediu esse problema irá validar suas propostas, definir um escopo fechado para futuramente não entrar em um ciclo infinito o que era para ser apenas uma base de dados.

Existem muitas formas de organizar um problema de negócio, mas não irei abordar mais esse assunto aqui, somente nas atividades, onde recomendo entender a pergunta, pois existem perguntas que requerem um relatório, ou seja, você escolhe qual ferramenta fazer esse relatório, e como você vai fazer, qual ferramenta vai utilizar, Excel, Word, vai fazer uma aplicação, vai enviar por e-mail, enfim, muitas possibilidades.

Como Manipular esses Dados

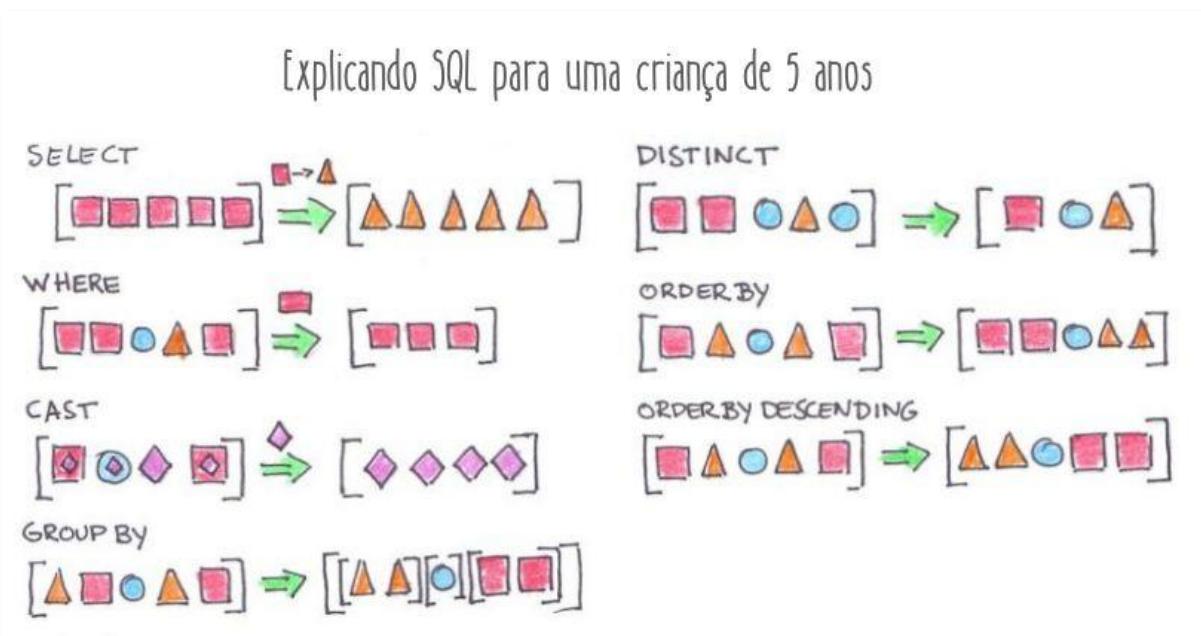


Figura 1: SQL Para Crianças, Imagem do LinkedIn, com algumas instruções importantes.

Para conseguir manipular esses dados, usamos um SGBD que é um software, com esse software para gerenciar nossas bases de dados, existem 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.

Possui diversos comandos, que são denominados como instruções, as instruções retornam à consulta, ou seja, os dados da minha base de dados. Para realizar uma consulta, antes é preciso entender as instruções, as instruções estão divididas em quatro principais grupos:

1. DML (Data Manipulation Language).

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

2. DDL (Data Definition Language).

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

3. DCL (Data Control Language).

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

4. DTL (Data Transaction Language).

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

Usamos os comandos DLL, geralmente em Objetos (Tabelas), tente não decorar esses termos, pois vamos trinhar os mesmos passo a passo ao decorrer do documento, vamos abordar necessariamente a sintaxe da linguagem SQL, que nada mais é que a forma de escrever o que você quer para que o computador entenda.

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 por exemplo.

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’.

Mas eu posso expressar de outra forma utilizando as famosas *varchars*.

Para o SQL, pode ter valores numerais e símbolos incluídos em uma sequência de caracteres que podem ser armazenados em variáveis, são um tipo diferente de dados, que juntam símbolos e valores decimais.

The screenshot shows a database interface with two main sections. On the left, a tree view displays the schema of a table named 'Address'. The 'Address' column is highlighted with a blue border. Other columns listed are City, Region, PostalCode, Country, HomePhone, Extension, Photo, Notes, ReportsTo, and PhotoPath. On the right, a grid view shows nine rows of data under the 'Address' column header. The data includes various addresses such as '507 - 20th Ave. E. Apt. 2A', '908 W. Capital Way', and '722 Moss Bay Blvd.'

	Address
1	507 - 20th Ave. E. Apt. 2A
2	908 W. Capital Way
3	722 Moss Bay Blvd.
4	4110 Old Redmond Rd.
5	14 Garrett Hill
6	Coventry House Miner Rd.
7	Edgeham Hollow Winchester Wa
8	4726 - 11th Ave. N.E.
9	7 Houndstooth Rd.

Figura 2: Informações e alguns registros na coluna *Address*, uma das colunas em uma tabela de dados.

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

Instruções geralmente dentro de parênteses são denominados **parâmetros**, provavelmente você já sabia disso quando estudou funções lá no ensino médio, caso não lembre, aqui vai um exemplo bem básico. $f(x) = x^2$.

Se eu mudar o x para dois, tenho a função $f(2) = 2^2$, e o dois está entre parênteses e onde tiver a letra x na minha função, altero para dois.

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 preencherá 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 capítulo, vai ser especificado esses tipos de dados que são muito importantes, principalmente as variáveis.

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 armazenar dados de vendas, realizar contas matemáticas, entre outras coisas.

	Unit Price	Units In Stock	Discontinued
1	18.00	39	0
2	19.00	17	0
3	10.00	13	0
4	22.00	53	0
5	21.35	0	1
6	25.00	120	0
7	30.00	15	0

Figura 3: Tipos de dados das colunas *UnitPrice*, *UnitsInStock*, e *Discontinued*.

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 capítulo, vamos primeiro entender os tipos numéricos usados na linguagem SQL.

Bit	Número inteiro de 0 ou 1 e nulo.
Int	É subdividido em TinyInt, SmallInt, Int e BigInt, o que muda é o espaço alocado 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 <i>float</i> , a variação é a precisão.

Tipos de dados Temporais

Um tipo de dado especialmente desenvolvido para armazenar valores do tipo de datas e tempos, que podem variar de diversas formas.

	OrderDate	RequiredDate	ShippedDate
1	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000	1996-07-16 00:00:00.000
2	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000	1996-07-10 00:00:00.000
3	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-12 00:00:00.000
4	1996-07-08 00:00:00.000	1996-08-05 00:00:00.000	1996-07-15 00:00:00.000
5	1996-07-09 00:00:00.000	1996-08-06 00:00:00.000	1996-07-11 00:00:00.000

Figura 3: Tipos de dados das colunas *OrderDate*, *RequiredDate* e *ShippedDate*.

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 Datetime2, porém 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, quando o valor é nulo, quer dizer que não tem nenhum registro nessa determinada coluna, muito comum durante o processo de junções de tabelas. Caso o parâmetro seja ‘Não Nulo’ ou ‘Nulo’, quer dizer que o formato da coluna aceita valores nulos de inputs ou inserção de dados, ou não aceita registros nulos.

O booleano, sendo seu 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 diferenciar 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 minúsculas e todas as palavras são, a maioria auto explicativas e diferenciáveis pelo SGBD.

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 eventualmente vão aparecer durante as consultas.

```
SELECT * FROM Employees  
ORDER BY FirstName
```

Figure 4: Exemplo de uma Instrução simples.

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 e seus devidos tipos de dados especiais.

The left pane shows the object browser for the 'dbo.Categories' table:

- Colunas:
 - CategoryID (PK, int, não nulo)
 - CategoryName (nvarchar(15), não nulo)
 - Description (ntext, nulo)
 - Picture (image, nulo)
- Chaves:
 - PK_Categories
- Restrições
- Gatilhos
- Índices

The right pane shows the results of a query:

	CategoryID	CategoryName
1	1	Beverages
2	2	Condiments
3	3	Confections
4	4	Dairy Products
5	5	Grains/Cereals
6	6	Meat/Poultry
7	7	Produce
8	8	Seafood

Figura 5: Chaves primárias e chaves estrangeiras.

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, a tabela Y usa e referencia a tabela X, quando a PK de uma tabela aparece em outra tabela, ela é chamada de *Foreign Key*, ou chave estrangeira.

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

Vai ser abordado a criação de base de dados e tabelas com os identificadores e chaves estrangeiras ao decorrer do documento.

Operadores Relacionais e Matemáticos

A linguagem SQL suporta alguns símbolos matemáticos, mas os principais são as condicionais, que usamos em consultas.

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 \neq	Diferente, essa condicional vai verificar se o valor X é diferente do valor Y ($X \neq Y$), nesses casos, retorna uma condição booleana (Sim ou Não). ($3 \neq 1$), 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 \geq	Esse conceito se aplica também a o menor ou igual , ele verifica se o valor X é maior ou idêntico ao valor Y. ($10 \leq 50$) Positivo, ($20 \geq 20$) 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 compararam resultados booleanos, (*True* ou *False*) nos operadores relacionais, seu resultado é um valor booleano, com esse

valor podemos utilizar operadores lógicos para que seja 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 resultados 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.

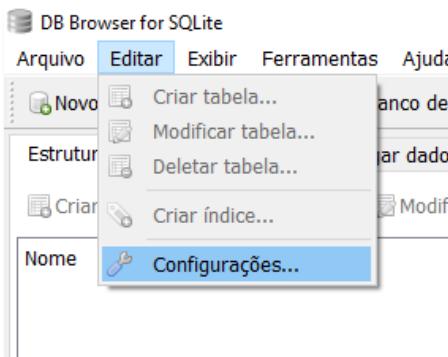


Figura 6: Guia de configurações no SQL Lite.

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

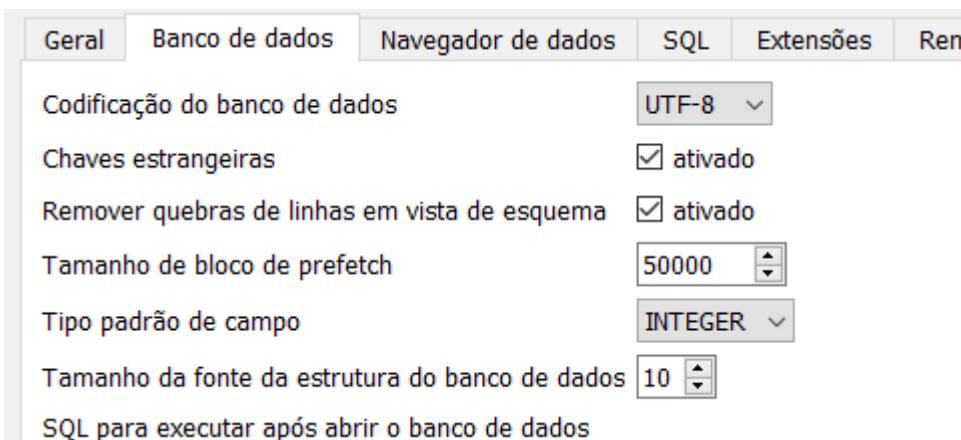


Figura 7: Configurações de SQL Lite.

Na guia Banco de Dados, certifique-se que a configuração está UTF-8, assim, você pode acrescentar acentos em seus registros.

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 ou no local padrão do SQLite, com algum nome.

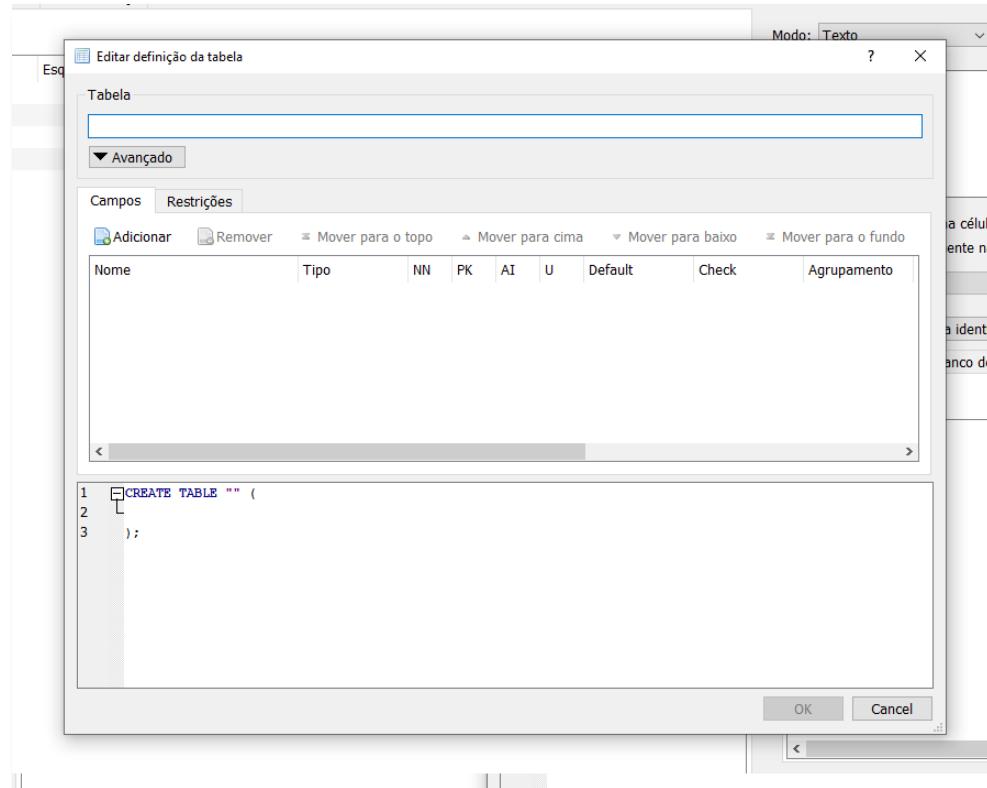


Figura 8: Criando uma tabela no SQL Lite.

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

Informações da nossa tabela:

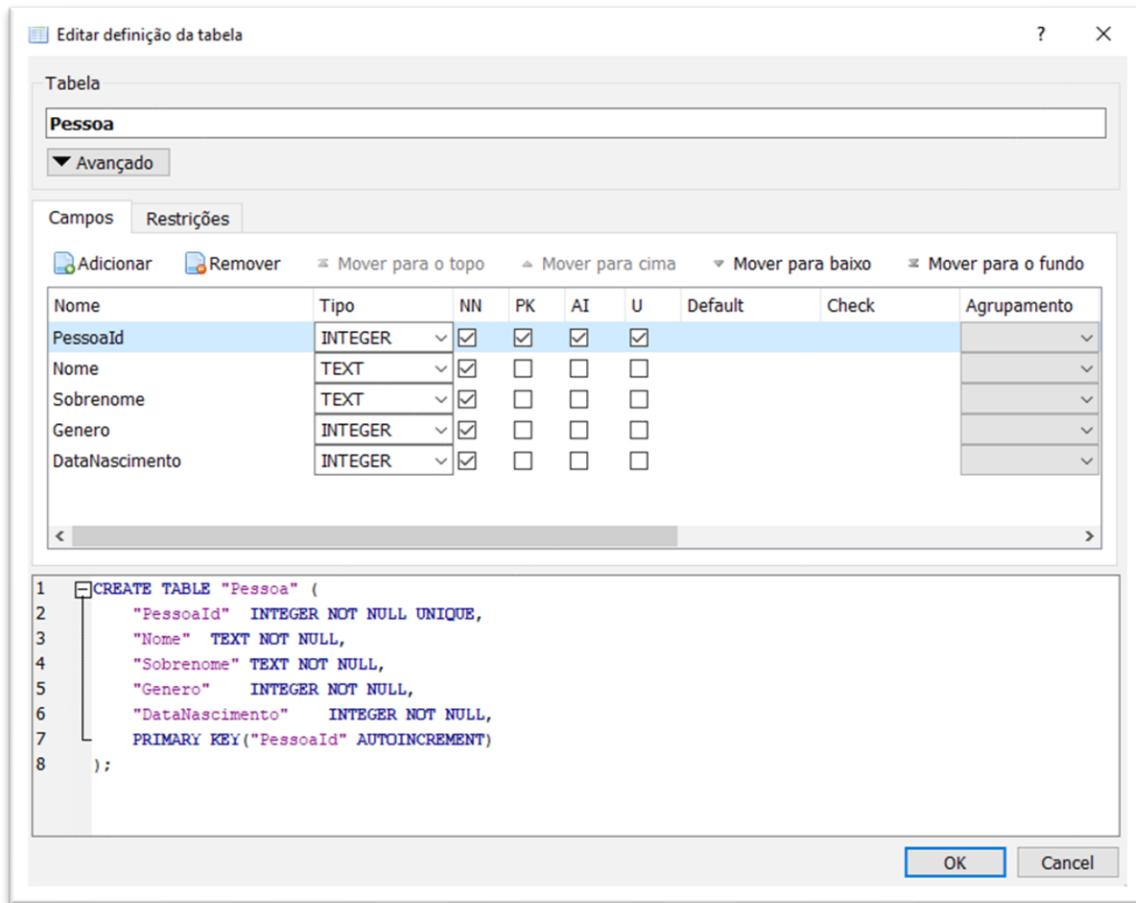


Figura 9: Criando colunas na tabela Pessoa.

No SQLite você consegue visualmente adicionar as colunas para começar criar 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 na instrução, tudo que foi checado, aparece ali também, e tudo que foi citado no primeiro tópico, fundamentos de SQL, 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, e sem acento.

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.

Entenda que, você escolhe como vai desenvolver, pois você é o analista, para isso deve atender aos requisitos do escopo que foi proposto pela unidade de negócio que lhe passou esse problema antes mesmo de começar a desenvolver a tabela. Só uma observação para não fazer exatamente igual ao meu exemplo, você tem total liberdade para desenvolver da forma que melhor atende a necessidade do negócio.

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.

Nome	Tipo	Esquema
PessoaId	INTEGER	"PessoaId" INTEGER NOT NULL UNIQUE
Nome	TEXT	"Nome" TEXT NOT NULL
Sobrenome	TEXT	"Sobrenome" TEXT NOT NULL
Genero	INTEGER	"Genero" INTEGER NOT NULL
DataNascimento	INTEGER	"DataNascimento" INTEGER NOT NULL
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Índices (0)		
Vistas (0)		
Gatilhos (0)		

Figura 10: Colunas da tabela Pessoa.

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

PessoaId	Nome	Sobrenome	Genero	DataNascimento
Filtro	Filtro	Filtro	Filtro	Filtro
1	1		0	0

Figura 11: Adicionando registros na tabela.

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

Alguns registros não nulos, o proprio SQLite vai preencher essas colunas.

The screenshot shows the DB Browser for SQLite interface. The title bar reads "DB Browser for SQLite - C:\Users\T-Gamer\Desktop\DB Browser for SQLite\PrimeiraBaseDeDados.db". The menu bar includes "Arquivo", "Editar", "Exibir", "Ferramentas", and "Ajuda". Below the menu is a toolbar with icons for "Novo banco de dados", "Abrir banco de dados", "Escrever modificações", "Reverter modificações", and "Abrir projeto". A tab bar at the top has tabs for "Estrutura do banco de dados", "Navegar dados", "Editar pragmas", and "Executar SQL". The main area is titled "Tabela: Pessoa" and displays a table with columns: Pessoaid, Nome, Sobrenome, Genero, and DataNascimento. There are three rows of data:

	Pessoaid	Nome	Sobrenome	Genero	DataNascimento
1	1	Joâna	Silvestre	1	20030523
2	2	José	Silva	1	19990901
3	3	Josélia	Silva	0	19980801

Figura 11: Registros manuais.

Podemos criar automaticamente as linhas na base de dados, basta clicar no ícone ‘Inserir um novo registro na tabela atual’ e você pode clicar nas linhas para adicionar as informações e digitar manualmente, após isso, basta clicar em aplicar.

Consultas Básicas

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

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

Sintax e Select

Lembra que no primeiro capítulo, 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.

The screenshot shows a SQLite database interface with a query window titled "SQL 1". The query entered is:

```

1 SELECT * FROM Pessoa
2

```

Below the query window is a results table with the following data:

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	1	20030523
2	2	José	Silva	0	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812

Figura 12: Adicionando registros na tabela.

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, porém é uma das mais perigosas, imagine-se em um cenário que você está manipulando uma base de dados imensa, caso usar um comando desse sem especificação, poderá travar a consulta ou até mesmo acabar com a memória do computador.

```

l  SELECT * FROM Pessoa LIMIT 2;


```

PessoalId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joāna Silvestre	1	20030523
2	2	José Silva	1	19990901

Execução finalizada sem erros.
 Resultado: 2 linhas retornadas em 4 ms
 Na linha 1:
 SELECT * FROM Pessoa LIMIT 2;

Figura 13: Instrução Limit.

Para conseguirmos reverter esse problema, existe uma instrução que podemos limitar essa consulta para retornar determinados registro, esse valor é arbitrário.

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

O ponto e vírgula ‘ ; ’ 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, essa instrução de distinguir funciona da mesma forma, a palavra em inglês para distinto é ‘*Distinct*’ , essa é a instrução no SQL.

The screenshot shows a SQL query window titled "SQL 1". The query entered is:

```
1 select distinct Sobrenome from PESSOA limit 4;
```

The results pane displays a table with one column "Sobrenome" containing two rows:

Sobrenome
1 Silvestre
2 Silva

Below the results, the status message reads:

Execução finalizada sem erros.
Resultado: 2 linhas retornadas em 1 ms
Na linha 1:
select distinct Sobrenome from PESSOA limit 4;

Figura 14: Instrução *Select* em minúsculo.

SQL não é case sensitive, ou seja, letras maiúsculas 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 encontrar valores que correspondem a essa condição, vai retornar o determinado registro.

The screenshot shows two side-by-side SQL query windows, both titled "SQL 1".

The left window contains the following query:

```
1 SELECT * FROM Pessoa
2 WHERE Nome = 'José'
```

The right window also contains the same query:

```
1 SELECT * FROM Pessoa
2 WHERE Nome = José
```

Both windows show the same result set, which is a table with columns: PessoaId, Nome, Sobrenome, Genero, DataNascimento. The data is:

PessoaId	Nome	Sobrenome	Genero	DataNascimento	
1	2	José	Silva	1	19990901

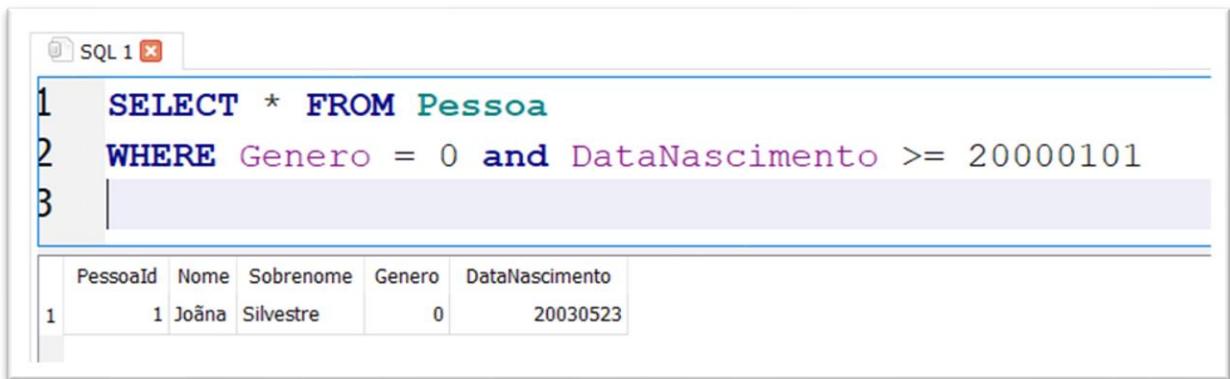
Figura 15: Instrução *Select* em maiúsculo com a instrução *Where*.

SELEÇÃO 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 nome José, não esta sendo representado com aspas, ainda especifiquemos durante a criação dessa tabela, que apenas aceitariamos registros do tipo string, e como foi visto no primeiro capítulo os tipos de dados, representa-se *String* com aspas.

Podemos usar os operadores lógicos durante as intruções WHERE.



The screenshot shows a SQL editor window titled "SQL 1". The query entered is:

```
1  SELECT * FROM Pessoa
2  WHERE Genero = 0 and DataNascimento >= 20000101
3  |
```

The result set is displayed below the query:

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Joãna	Silvestre	0	20030523

Figura 15: Instrução Select em maiúsculo com a instrução Where e And.

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 um tambem.

Palavra Reservada Like

Mas, caso eu não saiba exatamente o registro de uma tabela, então é usado a instrução **LIKE**, essa instrução, é geralmente utilizada em textos, quando não sabemos exatamente o que tinha escrito no texto, apenas uma parte, passamos junto uma mascara, o ‘%’, que é denominado padrão, existe muitos outros padroes, mas vou citar aqui apenas esse.

Esse padrão com as ‘%’, podem ser antes das aspas 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 com algum valor, porem terminam com o que eu digitei, mesma coisa para os outros casos.

The screenshot shows a SQL query window titled "SQL 1". The query is:

```

1  SELECT * FROM Pessoa
2  WHERE Nome LIKE '%ãn%'

```

The result set is a table with columns: PessoaId, Nome, Sobrenome, Genero, DataNascimento. There is one row:

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Joãna	Silvestre	0	20030523

Figura 16: Instrução *Select* em maiúsculo com a instrução *Where* e *Like*.

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

Palavra Reservada BETWEEN

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

The screenshot shows a SQL query window titled "SQL 1". The query is:

```

1  SeLeCt * FROM Pessoa
2  where PessoaId BeTWeEn 1 anD 4

```

The result set is a table with columns: PessoaId, Nome, Sobrenome, Genero, DataNascimento. There are four rows:

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Joãna	Silvestre	0	20030523
2	José	Silva	1	19990901
3	Josélia	Silva	0	19980801
4	Juca	Silvestre	1	17540812

Figura 17: Instrução *Select* em maiúsculo e minúsculo com a instrução *Where*, *Between* e *And*.

Reparam que, SQL, funciona com letras maiúsculas e minusculas juntos, mas não é muito agradável de ler 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 instrução **IN**, ‘Em’, passamso realmente uma lista de valores para serem selecionados.

```
1 SELECT * FROM Pessoa
2 WHERE PessoaId IN (1, 3, 4)
```

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Joána	Silvestre	0	20030523
2	Josélia	Silva	0	19980801
3	Juca	Silvestre	1	17540812

Figura 18: Instrução *Select* em maiúsculo com a instrução *Where* e *In*.

Selecione todas as pessoas da tabela pessoas, onde o id da pessoa 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 dessa linguagem é bem facil de entender e auto explicativas.

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.

```
SQL 1
1 SELECT * FROM Pessoa
2 WHERE Genero = 1
3 ORDER BY Nome DESC
```

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Marcos	Santos	1	20000101
2	Luiz	Lima	1	19850323
3	Juca	Silvestre	1	17540812
4	José	Silva	1	19990901

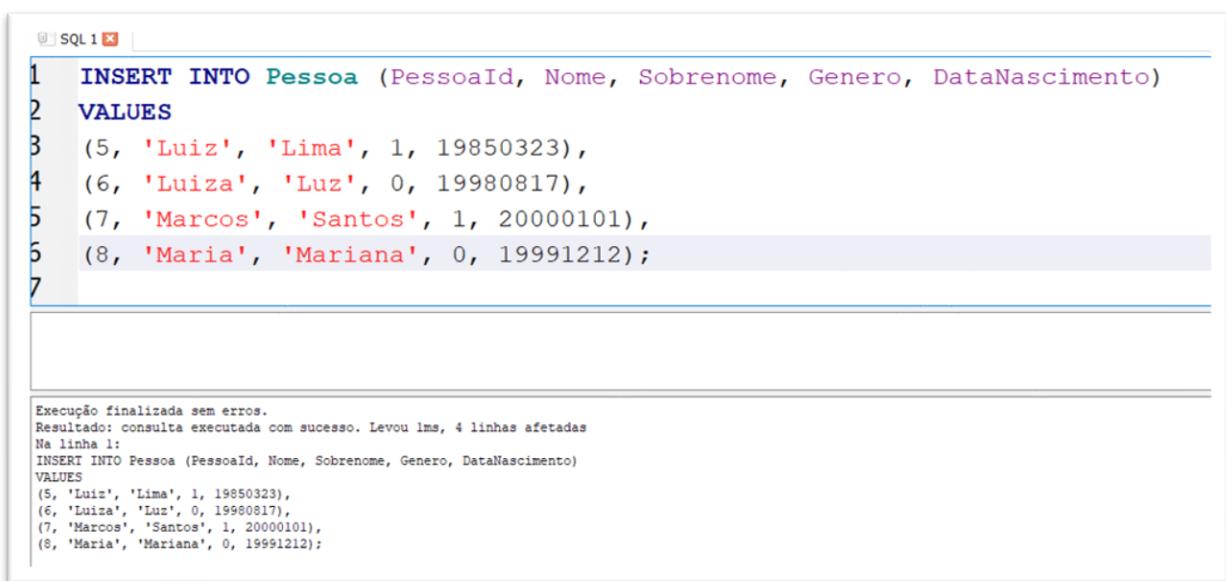
Figura 19: Instrução *Select* em maiúsculo com a instrução *Where*, *In Order By* e *Desc*.

A instrução de ordenação tambem funciona 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’.



The screenshot shows a SQL editor window titled "SQL 1". The code input area contains the following SQL statement:

```
1 INSERT INTO Pessoa (PessoaId, Nome, Sobrenome, Genero, DataNascimento)
2 VALUES
3 (5, 'Luiz', 'Lima', 1, 19850323),
4 (6, 'Luiza', 'Luz', 0, 19980817),
5 (7, 'Marcos', 'Santos', 1, 20000101),
6 (8, 'Maria', 'Mariana', 0, 19991212);
```

The output area shows the results of the execution:

```
Execução finalizada sem erros.
Resultado: consulta executada com sucesso. Levou 1ms, 4 linhas afetadas
Na linha 1:
INSERT INTO Pessoa (PessoaId, Nome, Sobrenome, Genero, DataNascimento)
VALUES
(5, 'Luiz', 'Lima', 1, 19850323),
(6, 'Luiza', 'Luz', 0, 19980817),
(7, 'Marcos', 'Santos', 1, 20000101),
(8, 'Maria', 'Mariana', 0, 19991212);
```

Figura 20: Instrução Select em maiúsculo com a instrução Where, In Order By e Desc.

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 vírgula e o ultimo valor com ponto e vírgula.

Depois veremos outras formas mais profissionais para realizar a inserção de registros em tabelas.

Palavra Reservada UPDATE e SET

Caso algum valor da minha tabela esteja errado, é possível atualizar somente aquele valor com instruções próprias da linguagem SQL. Para isso, usamos a instrução **UPDATE**, adicionei mais um valor a minha tabela com o gênero feminino, porém seu nome é ‘Guilherme’, claramente, esse é um erro input, que o próprio usuário pode ter selecionado errado ou o banco de dados armazenou errado.

The screenshot shows a SQLite database window titled "SQL 1". The SQL code entered is:

```

1 UPDATE Pessoa
2 SET Genero = 1
3 WHERE PessoaId = 9
4
5 SELECT * FROM PESSOA
6

```

Below the code, a table displays the data from the "PESSOA" table:

PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	Joãna	Silvestre	0	20030523
2	José	Silva	1	19990901
3	Josélia	Silva	0	19980801
4	Juca	Silvestre	1	17540812
5	Luiz	Lima	1	19850323
6	Luiza	Luz	0	19980817
7	Marcos	Santos	1	20000101
8	Maria	Mariana	0	19991212
9	Guilherme	Pereira	1	20010912

Figura 21: Instruções Selecionadas.

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, por isso essa instrução está até em negrito quando foi citada pela primeira vez.

Palavra Reservada DELETE

A instrução delete, como o nome já sugere, serve para deletar os registros da minha tabela, parecido com a função *update*, 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.

The screenshot shows a SQL editor window titled "SQL 1". It contains the following code:

```

1  DELETE FROM Pessoa
2  WHERE PessoaId = 9
3
4  SELECT * FROM Pessoa

```

Below the code, there is a table with 8 rows of data:

	PessoaId	Nome	Sobrenome	Genero	DataNascimento
1	1	Joãna	Silvestre	0	20030523
2	2	José	Silva	1	19990901
3	3	Josélia	Silva	0	19980801
4	4	Juca	Silvestre	1	17540812
5	5	Luiz	Lima	1	19850323
6	6	Luiza	Luz	0	19980817
7	7	Marcos	Santos	1	20000101
8	8	Maria	Mariana	0	19991212

Figura 22: Realizando todas as consultas.

Palavra Reservada DROP

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

The screenshot shows a SQL editor window titled "SQL 1". It contains the following code:

```

1  DROP TABLE Pessoa
2

```

Figura 23: 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 capítulo, 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, está 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.

Contexto de Negócio em Mercado

Sim, eu tinha falado anteriormente que não iria comentar mais sobre isso, mas é interessante você entender o que iremos fazer a partir daqui.

Os irmãos Jonas e Eduardo, dois sócios e amigos de empreendimento de sucesso e estão planejando entrar no mercado, mais precisamente no varejo, com apenas alguns produtos, porém mesmo com os produtos de entrada, a ideia inicial e a audiência na região selecionada para começar o desenvolvimento do empreendimento, eles não têm uma expertise para desenvolver uma base de dados básica para começar a essa Idea, então ambos contrataram uma equipe de consultoria para conseguir realizar esse desafio.

Para a equipe de consultoria, fizeram as seguintes perguntas.

- Preciso de uma base de dados para armazenar meus produtos.
- Preciso de várias tabelas, como Empregados, Produtos, Provedores
- É necessário popular com alguns dados que possuo e testar.
- Estavamos pensando sobre uma categoria externa para futuros produtos ou uma categoria já no produto?

Vou deixar sua criatividade livre para escolher o nome do mercado.

Lembram-se, realmente entenda o que eles querem que você faça, quebre o problema em partes e anote detalhadamente o projeto a ser concluído, assim você pode assinar o contrato e vai desenvolver exatamente o que foi proposto nas reuniões, etc.

Como não estamos inseridos no negócio, tente imaginar-se como você faria isso, você deve ir ao mercado certo, então, nos mercados possui o pessoal que trabalha, chamados de empregados ou colaboradores, possui categorias para os produtos, pois você não vai ver uma pasta de dentes junto com um exemplo de carne por exemplo.

Possui também os provedores ou fornecedores dos alimentos, fornecedores de equipamento geral, de produtos higiênicos, frutas, entre outros, todos esses exemplos vão ser as tabelas.

Fundamentos das Tabelas

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

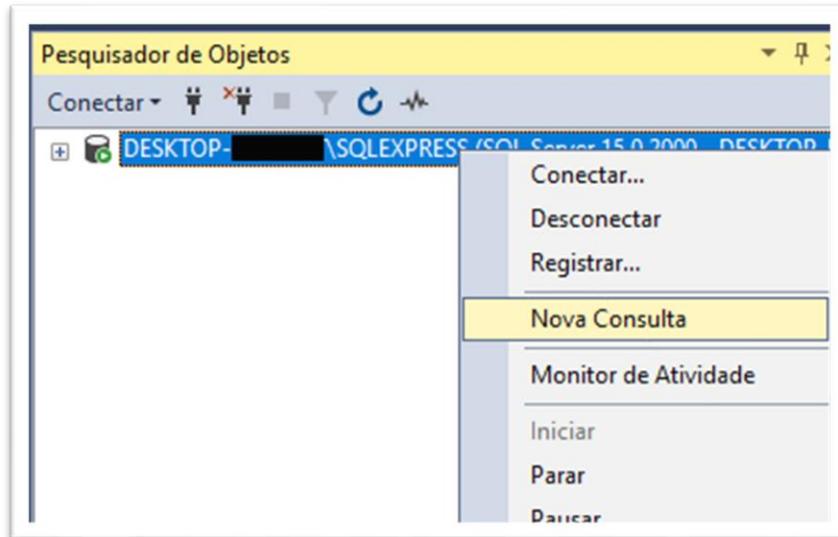


Figura 24: Iniciar uma consulta no SQL Server.

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.

CREATE DATABASE Market

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*, caso queira deletar a ‘Market’, basta clicar com o botão direito e em nova consulta e deletar.

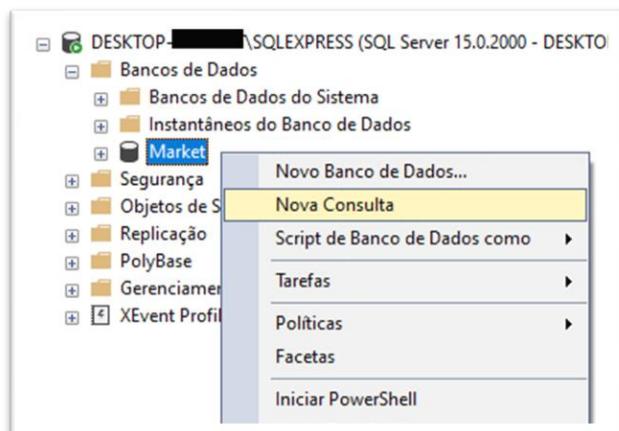


Figura 25: Realizar consultas em uma tabela específica.

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

Palavras Reservadas para Tabelas

```
CREATE TABLE Category (
    CategoryId  INT PRIMARY KEY NOT NULL,
    CategoryName VARCHAR(150) NOT NULL UNIQUE,
);

CREATE TABLE Supplier (
    SupplierId  INT PRIMARY KEY NOT NULL,
    CompanyName NVARCHAR(100) NOT NULL UNIQUE,
    ContactName NVARCHAR(50) NOT NULL,
    City        NVARCHAR(50) NOT NULL,
    Address     NVARCHAR(100) NOT NULL,
    Phone       NVARCHAR(20) NOT NULL
```

Figura 26: Criando uma tabela no SQL Server.

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.

```

CREATE TABLE Product (
    ProductId   INT PRIMARY KEY NOT NULL,
    ProductName VARCHAR(150) NOT NULL UNIQUE,
    CategoryId  INT FOREIGN KEY REFERENCES Category(CategoryId),
    Supplier     INT FOREIGN KEY REFERENCES Supplier(SupplierId),
    UnitPrice    MONEY NOT NULL,
);

CREATE TABLE Employee (
    EmployeeId  INT PRIMARY KEY NOT NULL,
    FirstName   NVARCHAR(50) NOT NULL,
    LastName    NVARCHAR(50) NOT NULL,
    Title       NVARCHAR(20) NOT NULL,
    Gender      NCHAR(2) CHECK (Gender in ('M', 'F')),
    BirthDate   DATETIME NOT NULL DEFAULT GETDATE(),
    HireDate    DATETIME NOT NULL DEFAULT GETDATE(),
    State       NVARCHAR(50) DEFAULT 'Ohio',
    Country     NVARCHAR(10) DEFAULT 'USA',
    City        NVARCHAR(100) DEFAULT 'Toledo',
    Photo       IMAGE,
    Notes       NTEXT,
);

```

Figura 27: Criando mais tabelas para a base de dados Market.

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 contém várias informações que já foram citadas, porém 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 lógica do default foi selecionada e aplicada nas próximas colunas, para obter valores *default*.

A coluna foto, o valor de seu registro é binário, porém 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’.

Pois, Jonas e Eduardo querem saber algumas coisas a mais, caso você tenha fechado o escopo como citei anteriormente, pode prosseguir caso quebrou o problema em partes e realmente conversou e entendeu as causas do problema, ou seja, fechando assim o escopo do projeto, ou assinando a consultoria.

```
CREATE TABLE Customer (
    CustomerId INT PRIMARY KEY NOT NULL,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    State NVARCHAR(50) DEFAULT 'Ohio',
    Country NVARCHAR(10) DEFAULT 'USA',
);

CREATE TABLE Orders (
    OrdersId INT PRIMARY KEY NOT NULL,
    CustomerId INT FOREIGN KEY REFERENCES Customer(CustomerId),
    EmployeeId INT FOREIGN KEY REFERENCES Employee(EmployeeId),
    OrderDate DATETIME NOT NULL DEFAULT GETDATE(),
    ShippingDate DATETIME NOT NULL DEFAULT GETDATE()
);

CREATE TABLE OrderDetail (
    OrdersId INT PRIMARY KEY NOT NULL,
    ProductId INT FOREIGN KEY REFERENCES Orders(OrdersId),
    Quantity SMALLINT NOT NULL,
    UnitPrice MONEY NOT NULL,
);
```

Figura 28: Mais algumas tabelas com seus devidos tipos de dados.

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 várias funcionalidades, a primeira e mais simples, é a instrução para deletar uma tabela, vamos usar as palavras reservadas, **Alter**, **Table**, **Drop** e **Constraint**.



Figura 29: Deletar uma chave estrangeira de uma tabela.

Figura 31: Pasta chaves, onde está todas as informações de chaves estrangeiras e primárias da tabela.

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 para não ter que digitar todo o código.

Depois de desanexar a chave, vamos dropar a tabela com a instrução *drop column* e passando a tabela em específico 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, por isso, crie com cuidado suas tabelas.

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*, porém, temos que especificar o tipo de dados que essa coluna vai aceitar, no exemplo, usei o tipo inteiro.



Figura 32: Adicionando coluna na tabela.

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

Estou alterando a tabela *Product* e adicionando uma coluna nova chamada *SupplierId* do tipo INT.

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.

```
ALTER TABLE Product  
ALTER COLUMN SupplierId BIT
```

Figura 33: Alterando uma coluna específica da tabela.

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

```
ALTER TABLE Product  
ADD SupplierId INT FOREIGN KEY REFERENCES Supplier(SupplierId)
```

Resultados Mensagens

ProductId	ProductName	UnitPrice	CategoryId	SupplierId

Figura 34: Adicionando chave estrangeira em uma coluna.

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

```

ALTER TABLE Product
ADD EmployeeId INT

ALTER TABLE Product
ADD CONSTRAINT EmployeeId FOREIGN KEY (EmployeeId)
REFERENCES Employee(EmployeeId)

```

	ProductId	ProductName	UnitPrice	CategoryId	SupplierId	EmployeeId
1	1	Shovel	60.00	6	3	NULL

Figura 35: Adicionando mais chaves estrangeiras.

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.

```

SELECT MAX(UnitPrice) as MaxValue FROM Product
SELECT MIN(UnitPrice) as MinValue FROM Product
SELECT SUM(UnitPrice) as SumValue FROM Product
SELECT COUNT(ProductId) as Quantity FROM Product

SELECT AVG(UnitPrice) as Mean FROM Product
SELECT SUM(UnitPrice) / COUNT(ProductId) FROM Product

SELECT DISTINCT ROUND(SQRT(UnitPrice), 2) as SqrtPrice FROM Product

```

	MaxValue
1	75.00

Figura 36: Algumas funções em SQL Server.

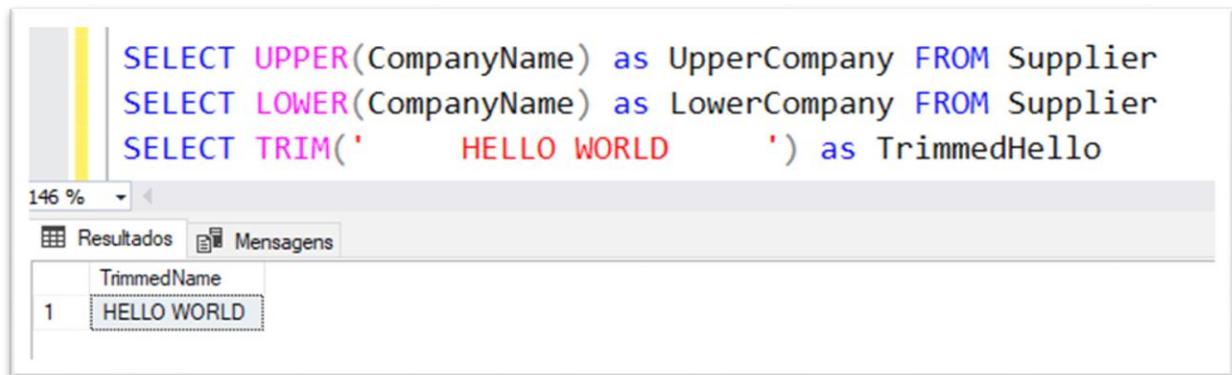
As funções são palavras reservadas, que recebem como parâmetro as colunas que você quer realizar as instruções.

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 última instrução, vai retornar os valores *únicos*, onde estou aplicando a função **SQRT**, que retorna a raiz quadrada, porem o resultado dessa função está em outra função, a **ROUND**, essa instrução, reduz os caracteres depois da vírgula, como muitos valores da raiz quadrada vem com vírgula, foi aplicado uma função de arredondamento, passando o valor para ser arredondado e a quantidade de casas após a vírgula, 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.



```
SELECT UPPER(CompanyName) as UpperCompany FROM Supplier
SELECT LOWER(CompanyName) as LowerCompany FROM Supplier
SELECT TRIM('    HELLO WORLD      ') as TrimmedHello
```

TrimmedName
1 HELLO WORLD

Figura 37: Mais algumas funções.

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

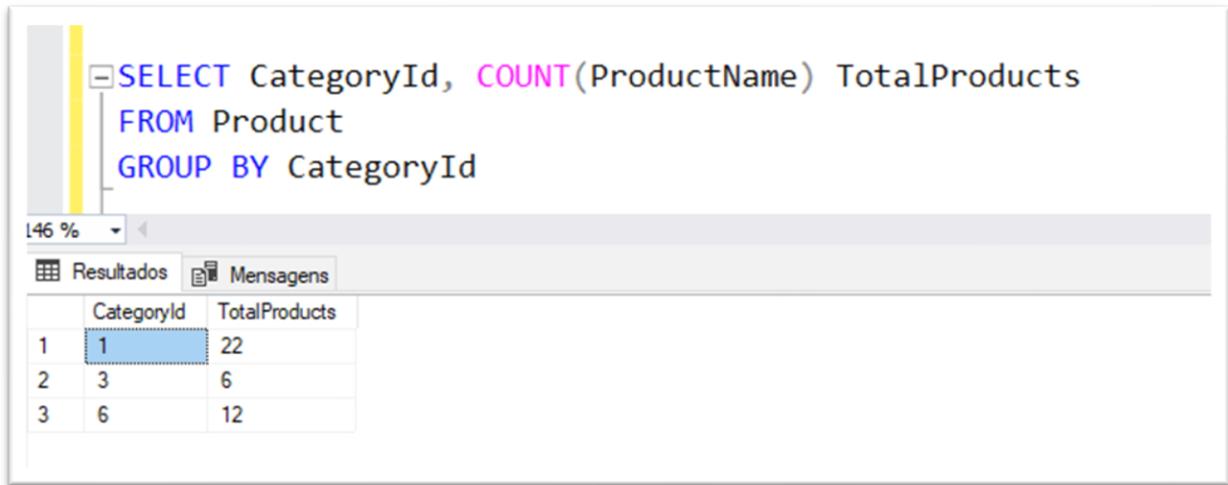
A função **UPPER**, converte todos os registros passados como parâmetros para maiúsculo.

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

A função **TRIM**, é a mesma da estatística, sua tradução é ‘apurar’, 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 é 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.



```
SELECT CategoryId, COUNT(ProductName) TotalProducts
FROM Product
GROUP BY CategoryId
```

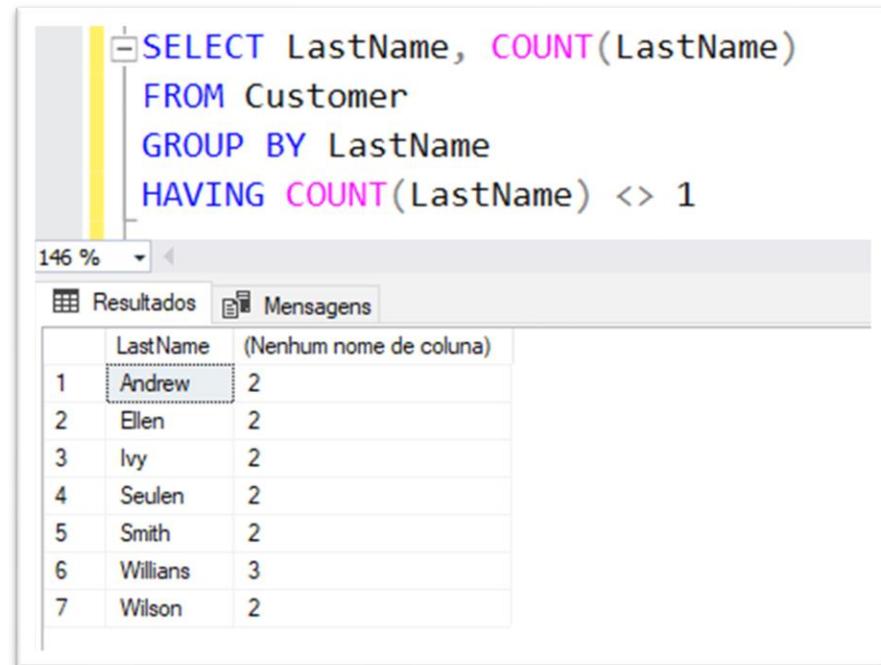
The screenshot shows an SQL query in the top pane and its results in the bottom pane. The results table has two columns: CategoryId and TotalProducts. The data is:

CategoryId	TotalProducts
1	22
3	6
6	12

Figura 37:Simples agrupamentos de valores.

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*.



```
SELECT LastName, COUNT(LastName)
FROM Customer
GROUP BY LastName
HAVING COUNT(LastName) <> 1
```

The screenshot shows an SQL query in the top pane and its results in the bottom pane. The results table has two columns: LastName and (Nenhum nome de coluna). The data is:

LastName	(Nenhum nome de coluna)
Andrew	2
Ellen	2
Ivy	2
Seulen	2
Smith	2
Willians	3
Wilson	2

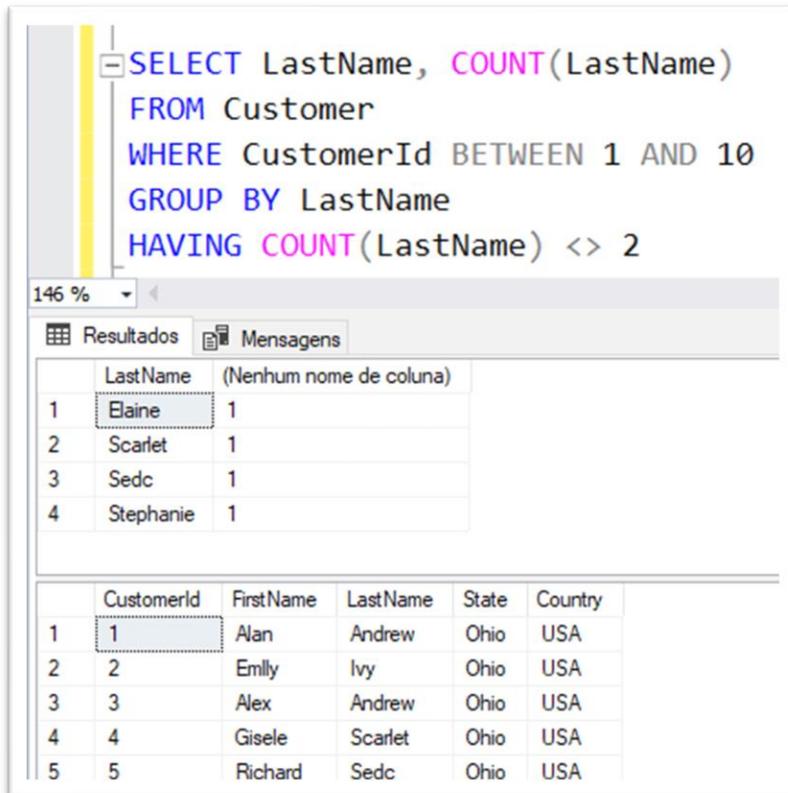
Figura 38: Instrução *Having*.

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 capítulo desse guia, tem a lista de referências, e lá tem um dos sites que obtive algumas referências para colocar na base de dados.



The screenshot shows a SQL query window with the following code:

```
SELECT LastName, COUNT(LastName)
FROM Customer
WHERE CustomerId BETWEEN 1 AND 10
GROUP BY LastName
HAVING COUNT(LastName) <> 2
```

The results pane displays two tables. The first table, titled "Resultados", shows the count of customers by last name where the count is not equal to 2:

	LastName	(Nenhum nome de coluna)
1	Elaine	1
2	Scarlet	1
3	Sedc	1
4	Stephanie	1

	CustomerId	FirstName	LastName	State	Country
1	1	Alan	Andrew	Ohio	USA
2	2	Emilly	Ivy	Ohio	USA
3	3	Alex	Andrew	Ohio	USA
4	4	Gisele	Scarlet	Ohio	USA
5	5	Richard	Sedc	Ohio	USA

Figura 39: Instrução *Having* com *GroupBy*.

Podemos sem problema utilizar instrução *Where*, para realizar consultas mais sigilosas. Nesse caso, ele fica antes do *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.

```

SELECT ProductName, UnitPrice, SupplierId FROM Product
WHERE UnitPrice > 10.00
AND SupplierId =
(
    SELECT SupplierId
    FROM Supplier
    WHERE CompanyName = 'Karkki Oy'
)
ORDER BY UnitPrice DESC

```

The screenshot shows a SQL query in the query editor. The query uses a subquery to filter products from a supplier whose company name is 'Karkki Oy'. The results are ordered by unit price in descending order. The results grid shows four rows:

	ProductName	UnitPrice	SupplierId
1	Pickaxe	75.00	3
2	Shovel	60.00	3
3	Board 10m	25.00	3
4	Huep I	20.00	3

Figura 40: Primeira SubQuery.

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.

Figura 40: Primeira SubQuery.

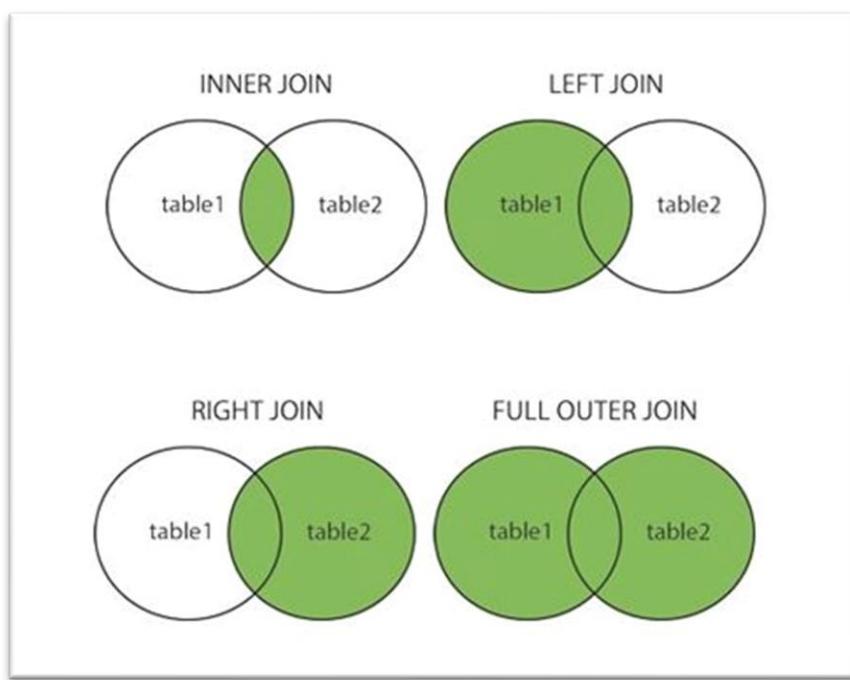


Figura 41: Gráfico de Junções.

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.

O conceito de correlação é bastante extenso, não vou abordar esse método estatístico.

Resultados	Mensagens																														
<pre>SELECT * FROM Product p, Product p2</pre>																															
<table border="1"> <thead> <tr> <th>ProductId</th> <th>ProductName</th> <th>UnitPrice</th> <th>CategoryId</th> <th>SupplierId</th> <th>ProductId</th> <th>ProductName</th> <th>UnitPrice</th> <th>CategoryId</th> <th>SupplierId</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Shovel</td> <td>60.00</td> <td>6</td> <td>3</td> <td>1</td> <td>Shovel</td> <td>60.00</td> <td>6</td> <td>3</td> </tr> <tr> <td>2</td> <td>Pickaxe</td> <td>75.00</td> <td>6</td> <td>3</td> <td>1</td> <td>Shovel</td> <td>60.00</td> <td>6</td> <td>3</td> </tr> </tbody> </table>	ProductId	ProductName	UnitPrice	CategoryId	SupplierId	ProductId	ProductName	UnitPrice	CategoryId	SupplierId	1	Shovel	60.00	6	3	1	Shovel	60.00	6	3	2	Pickaxe	75.00	6	3	1	Shovel	60.00	6	3	
ProductId	ProductName	UnitPrice	CategoryId	SupplierId	ProductId	ProductName	UnitPrice	CategoryId	SupplierId																						
1	Shovel	60.00	6	3	1	Shovel	60.00	6	3																						
2	Pickaxe	75.00	6	3	1	Shovel	60.00	6	3																						

Figura 42: Selecionando duas vezes da mesma coluna.

A priori é meio confuso, mas entenda agora que, o nosso sgbd, fez todas as possíveis combinações da nossa tabela, para ela mesmo, minhas 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.

The screenshot shows a SQL query window with the following code:

```
SELECT
    p.ProductId, p.ProductName, p.UnitPrice,
    p2.ProductId, p2.ProductName, p2.UnitPrice
FROM Product p, Product p2
WHERE p.UnitPrice = p2.UnitPrice
ORDER BY p.UnitPrice DESC
```

Below the query window is a results grid titled "Resultados". The grid contains 15 rows of data, each with two sets of product information (ProductId,ProductName,UnitPrice) corresponding to a single price value. The columns are labeled ProductId, ProductName, UnitPrice, ProductId, ProductName, and UnitPrice respectively.

	ProductId	ProductName	UnitPrice	ProductId	ProductName	UnitPrice
1	2	Pickaxe	75,00	2	Pickaxe	75,00
2	1	Shovel	60,00	1	Shovel	60,00
3	5	Board 10m	25,00	5	Board 10m	25,00
4	12	Hose L	20,00	12	Hose L	20,00
5	17	Bean 20C	20,00	12	Hose L	20,00
6	18	Bean 30C	20,00	12	Hose L	20,00
7	12	Hose L	20,00	17	Bean 20C	20,00
8	17	Bean 20C	20,00	17	Bean 20C	20,00
9	18	Bean 30C	20,00	17	Bean 20C	20,00
10	12	Hose L	20,00	18	Bean 30C	20,00
11	17	Bean 20C	20,00	18	Bean 30C	20,00
12	18	Bean 30C	20,00	18	Bean 30C	20,00
13	4	Board 05m	15,00	11	Hose M	15,00
14	11	Hose M	15,00	11	Hose M	15,00
15	24	Sap Ub Large	15,00	11	Hose M	15,00

Figura 42: Selecionando duas vezes da mesma coluna com mais instruções.

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.

	ProductId	ProductName	UnitPrice	CategoryId	SupplierId
1	1	Shovel	60,00	6	3
2	2	Pickaxe	75,00	6	3

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.

```

SELECT TOP 10
    p.ProductId, p.ProductName, p.UnitPrice,
    p.SupplierId,
    c.CategoryName
FROM Product p
    INNER JOIN Category c
        ON c.CategoryId = p.CategoryId
ORDER BY p.UnitPrice

```

ProductId	ProductName	UnitPrice	SupplierId	CategoryName
1	Wrench K	2,00	3	Geral Products
2	Sap Small	2,00	2	Hygienic Products
3	Sap Ub Small	2,00	2	Hygienic Products
4	Toothpaste Balkan	2,00	2	Hygienic Products
5	Toothpaste Island	3,00	2	Hygienic Products
6	Sap Sand Small	3,00	2	Hygienic Products

Figura 43: Consultas com *Inner Join*.

Similarmente ao *self 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.

The screenshot shows a SQL query window with the following code:

```
SELECT TOP 10
    p.ProductId, p.ProductName, p.UnitPrice,
    s.CompanyName,
    c.CategoryName
FROM Product p
    INNER JOIN Category c
        ON c.CategoryId = p.CategoryId
    INNER JOIN Supplier s
        ON s.SupplierId = p.SupplierId
ORDER BY p.UnitPrice DESC
```

The results grid displays the following data:

	ProductId	ProductName	UnitPrice	CompanyName	CategoryName
1	2	Pickaxe	75,00	Karkki Oy	Geral Products
2	1	Shovel	60,00	Karkki Oy	Geral Products
3	5	Board 10m	25,00	Karkki Oy	Geral Products
4	12	Hose L	20,00	Karkki Oy	Geral Products
5	17	Bean 20C	20,00	Gai pâturage	Cereals
6	18	Bean 30C	20,00	Gai pâturage	Cereals
7	11	Hose M	15,00	Karkki Oy	Geral Products
8	4	Board 05m	15,00	Karkki Oy	Geral Products
9	24	Soap Ub Large	15,00	Zaanse Snoepfabriek	Hygienic Products
10	16	Rice 30C	13,00	Gai pâturage	Cereals

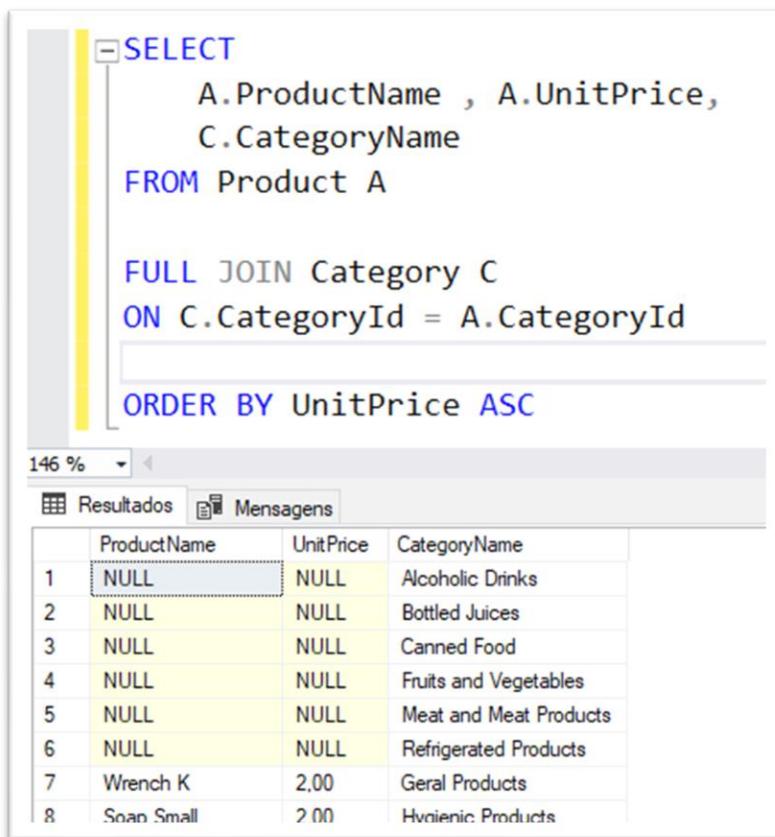
Figura 44: Consultas com mais de um *Inner Join*.

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.



The screenshot shows a SQL query window with the following code:

```
SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Product A
    FULL JOIN Category C
    ON C.CategoryId = A.CategoryId
ORDER BY UnitPrice ASC
```

The results pane shows the following data:

	ProductName	UnitPrice	CategoryName
1	NULL	NULL	Alcoholic Drinks
2	NULL	NULL	Bottled Juices
3	NULL	NULL	Canned Food
4	NULL	NULL	Fruits and Vegetables
5	NULL	NULL	Meat and Meat Products
6	NULL	NULL	Refrigerated Products
7	Wrench K	2,00	Geral Products
8	Soan Small	2,00	Hygienic Products

Figura 45: Consultas com mais de um *Inner Join*.

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.

Porém, 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 Right 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 foreign key.

```

SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Category C
RIGHT JOIN Product A
ON C.CategoryId = A.CategoryId
ORDER BY UnitPrice ASC

```

	ProductName	UnitPrice	CategoryName
1	Wrench K	2.00	Geral Products
2	Soap Small	2.00	Hygienic Products
3	Soap Ub Small	2.00	Hygienic Products
4	Toothpaste Balkan	2.00	Hygienic Products
5	Toothpaste Island	3.00	Hygienic Products
6	Soap Sand Small	3.00	Hygienic Products
7	Wrench K	2.00	Geral Products


```

SELECT
    A.ProductName , A.UnitPrice,
    C.CategoryName
FROM Category C
LEFT JOIN Product A
ON C.CategoryId = A.CategoryId
ORDER BY UnitPrice ASC

```

	ProductName	UnitPrice	CategoryName
1	NULL	NULL	Meat and Meat Products
2	NULL	NULL	Canned Food
3	NULL	NULL	Fruits and Vegetables
4	NULL	NULL	Bottled Juices
5	NULL	NULL	Alcoholic Drinks
6	NULL	NULL	Refrigerated Products
7	Wrench K	2.00	Geral Products
8	Soap Small	2.00	Hygienic Products

Figura 46: Consultas com *Right Join* e *Left Join*.

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 de dados 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.

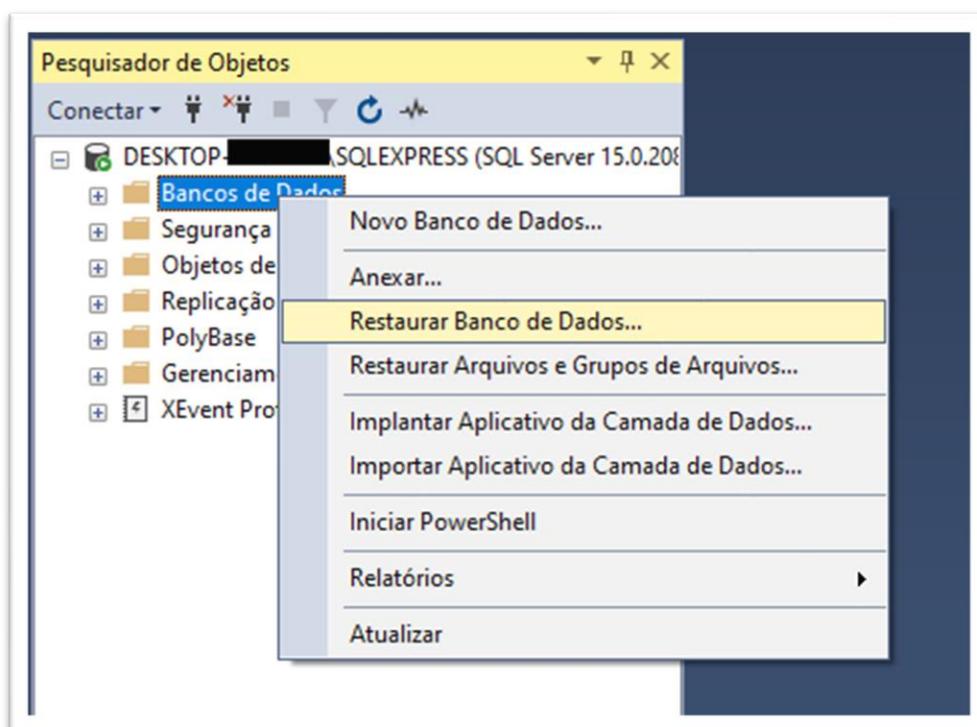


Figura 47: Restaurar uma base de dados.

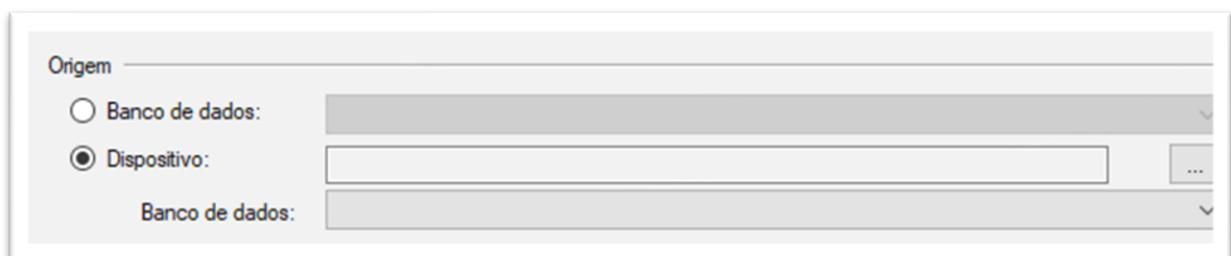


Figura 48: Informações da localização da base de dados.

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 clicar no botão adicionar.

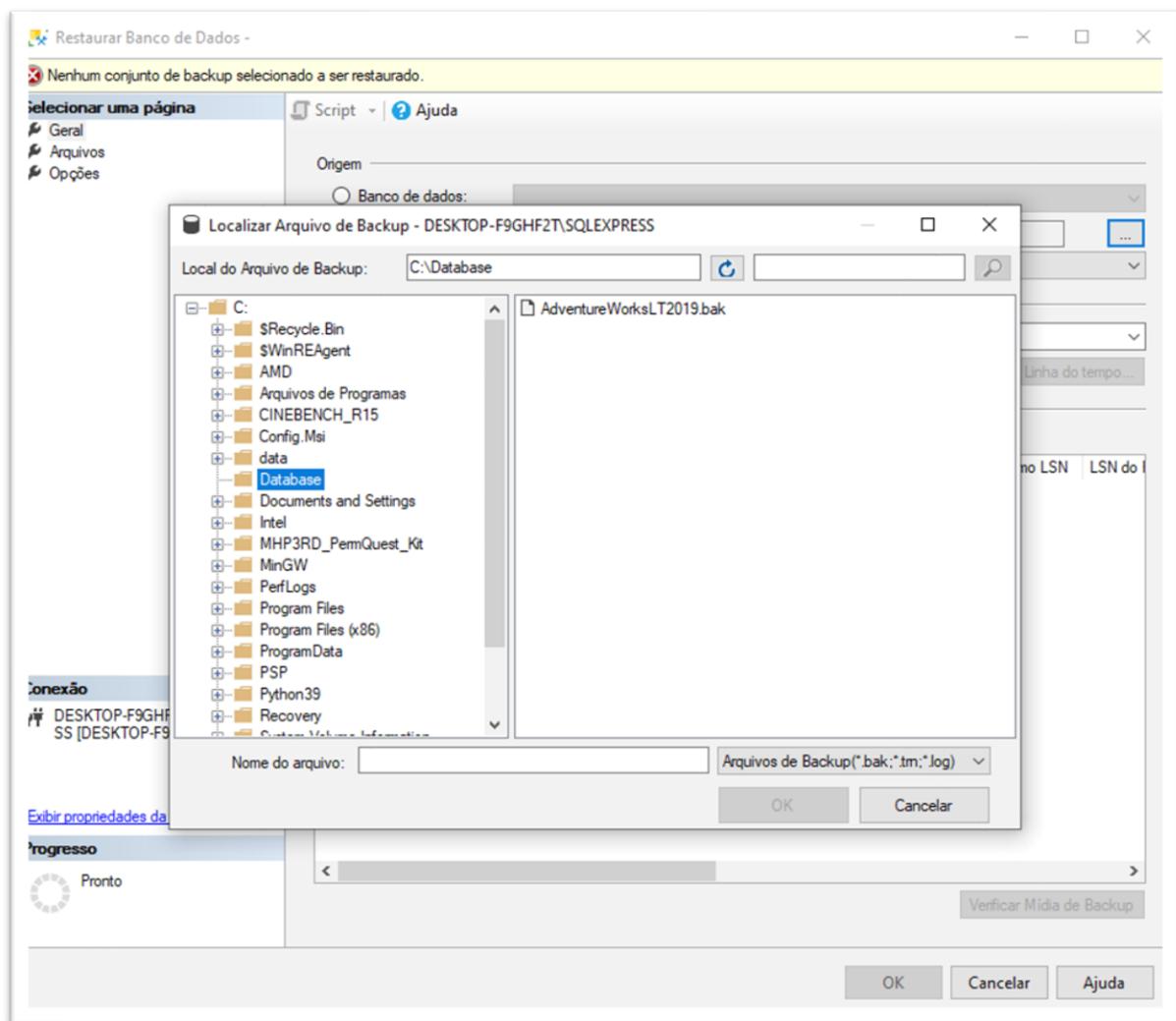


Figura 49: Selecionando o arquivo para restauração.

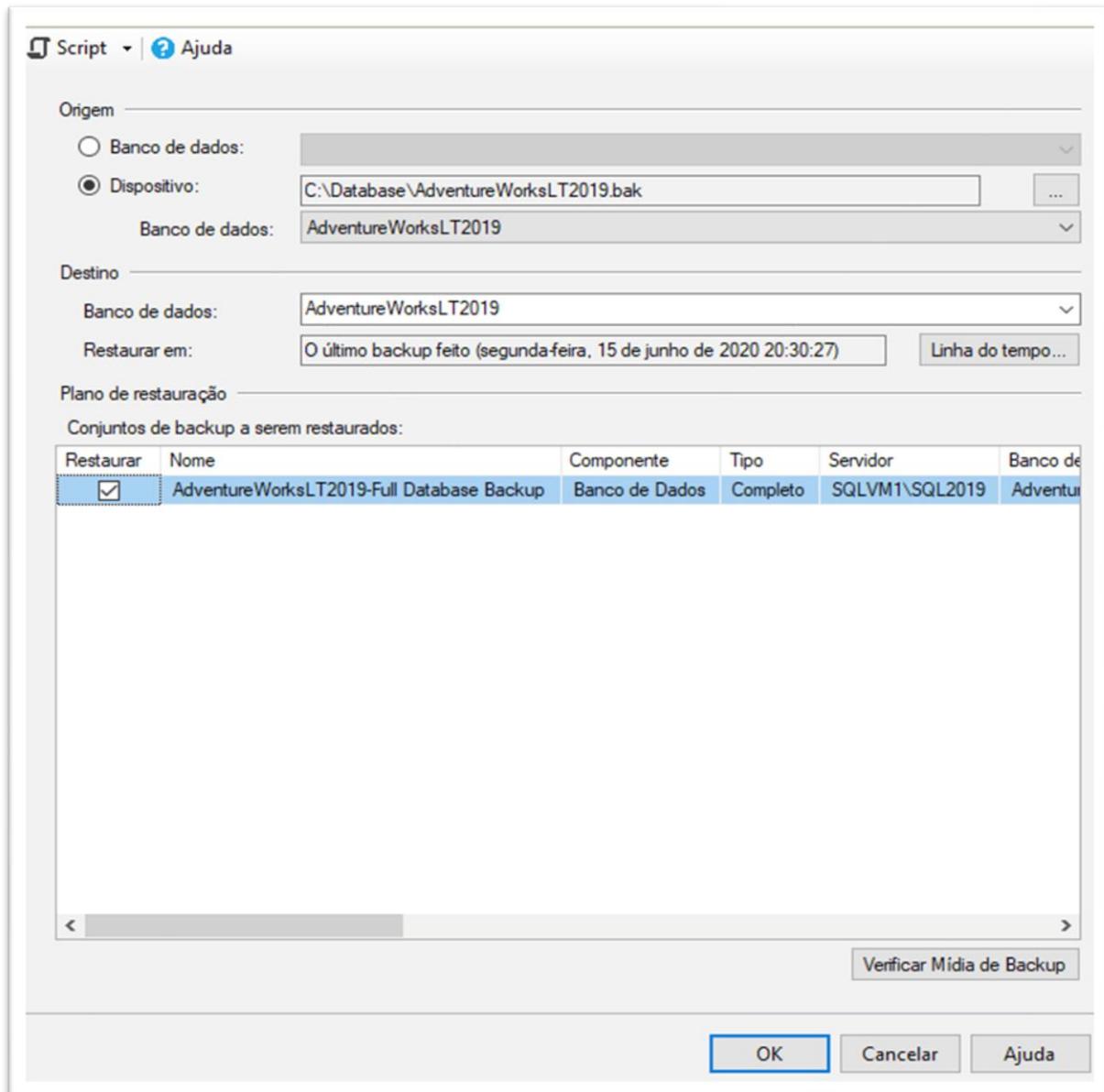


Figura 50: Preparando a restauração.

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.

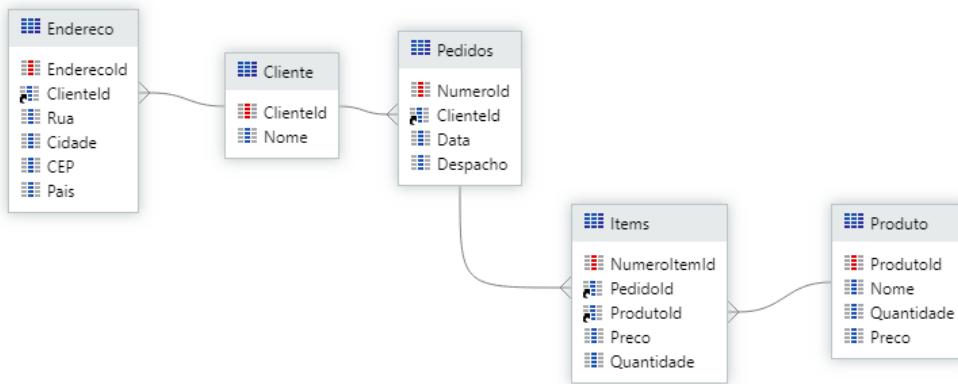


Figura 51: Esquema de Base de dados básica.

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.

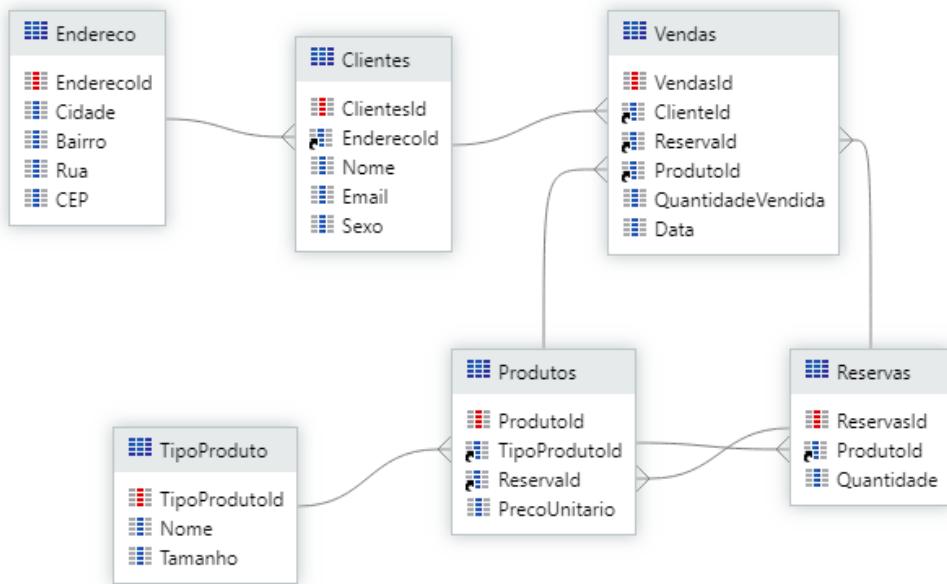


Figura 52: Outro Esquema de Base de dados básica.

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.

Scripts: Entenda como um arquivo originado de uma linguagem de programação que possui uma extensão, exemplo, arquivos de C++, possuem a extensão “nome.cpp”, ou arquivos Java Script, possuem a extensão “nome.js”.

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 (W3Schools) -> 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_sqldatabase.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-Fundamentals/tree/main/Atividades/SQL%20%26%20SQLite%20Fundamentos>

Lista de Atividades SQL Server -> <https://github.com/xGabrielR/SQL-Fundamentals/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*