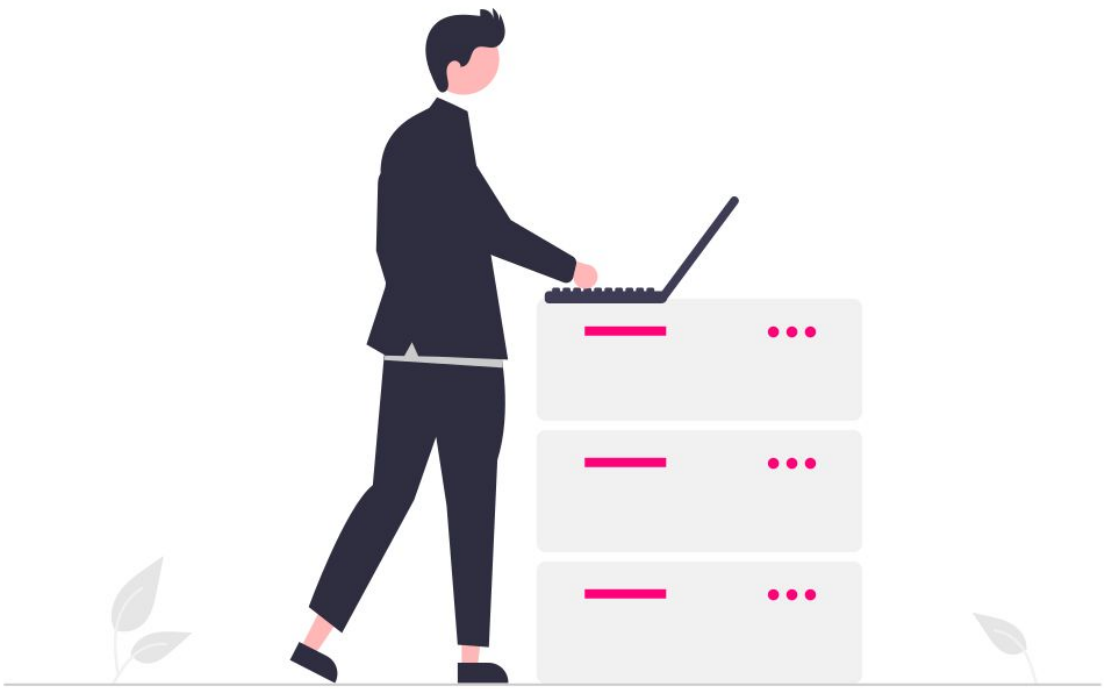
SQL



Realizando Consultas

Realizando Consultas

SELECT

O comando **SELECT** é uma **palavra reservada** do **SQL** utilizada para **construir queries** de consultas em banco de dados.

Chamamos de **query** um **trecho de código SQL** que representa uma instrução a ser executada no banco de dados.

A instrução abaixo, representa uma query para consultar todos os registros da tabela **products**.

```
select * from products;
```

A query apresentada anteriormente, ao ser executada, retorna todos os registros da tabela **products** conforme tabela abaixo.

ID	EAN	TITLE	CATEGORY	VENDOR	PRICE	RATING	CREATED_AT
1	1018947080336	Rustic Paper Wallet	Gizmo	Swaniawski, Casper and Hilll	29,46	4,6	julho 19, 2017, 7:44 PM
2	7663515285824	Small Marble Shoes	Doohickey	Balistreri-Ankunding	70,08	0	abril 11, 2019, 8:49 AM
3	4966277046676	Synergistic Granite Chair	Doohickey	Murray, Watsica and Wunsch	35,39	4	setembro 8, 2018, 10:03 PM
4	4134502155718	Enormous Aluminum Shirt	Doohickey	Regan Bradtke and Sons	73,99	3	março 6, 2018, 2:53 AM
5	5499736705597	Enormous Marble Wallet	Gadget	Price, Schultz and Daniel	82,75	4	outubro 3, 2016, 1:47 AM
6	2293343551454	Small Marble Hat	Doohickey	Nolan-Wolff	64,96	3,8	março 29, 2017, 5:43 AM
7	0157967025871	Aerodynamic Linen Coat	Doohickey	Little-Pagac	98,82	4,3	junho 3, 2017, 3:07 AM
8	1078766578568	Enormous Steel Watch	Doohickey	Senger-Stamm	65,89	4,1	abril 30, 2018, 3:03 PM

Realizando Consultas

SELECT coluna1, coluna2...

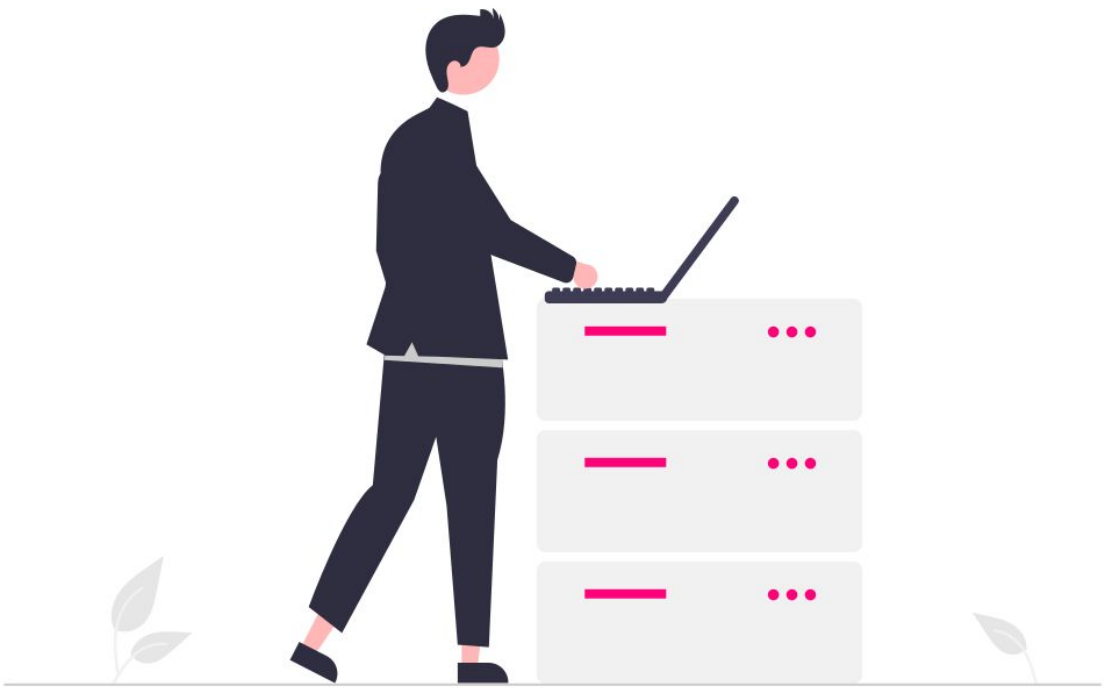
Eventualmente, desejamos retornar apenas alguns campos em uma consulta. Observe a instrução abaixo:

```
select id, title, category, price from products;
```

Nesse caso, ao executar a query acima, será retornado os campos **id, title, category e price** de todos os registros da tabela **products**.

ID	TITLE	CATEGORY	PRICE
1	Rustic Paper Wallet	Gizmo	29,46
2	Small Marble Shoes	Doohickey	70,08
3	Synergistic Granite Chair	Doohickey	35,39
4	Enormous Aluminum Shirt	Doohickey	73,99
5	Enormous Marble Wallet	Gadget	82,75
6	Small Marble Hat	Doohickey	64,96
7	Aerodynamic Linen Coat	Doohickey	98,82
8	Enormous Steel Watch	Doohickey	65,89

SQL



Filtrando Registros

Filtrando Registros

WHERE

Os comandos utilizados para manipulação de dados com SQL são chamados de **cláusulas**. A cláusula **WHERE** é usada para **filtrar registros** que atendem a uma ou várias **condições** e essas condições são validadas através de **expressões lógicas**.

Observe a instrução abaixo:

```
select * from products where price < 30;
```

A query apresentada anteriormente, ao ser executada, retorna todos os registros que atendem a condição solicitada. Nesse caso, onde o campo **price** de cada registro seja **menor que 30**.

ID	EAN	TITLE	CATEGORY	VENDOR	PRICE	RATING	CREATED_AT
1	1018947080336	Rustic Paper Wallet	Gizmo	Swaniawski, Casper and Hilll	29,46	4,6	julho 19, 2017, 7:44 PM
14	8833419218504	Awesome Concrete Shoes	Widget	McClure-Lockman	25,1	4	dezembro 31, 2017, 2:41 PM
15	5881647583898	Aerodynamic Paper Computer	Widget	Friesen-Anderson	25,1	4	setembro 8, 2016, 2:42 PM
20	5099742600901	Intelligent Wooden Gloves	Gizmo	Ora Monahan and Sons	24,88	4	agosto 25, 2017, 6:18 PM
22	7595223735110	Enormous Marble Shoes	Gizmo	Mayer, Kiehn and Turcotte	21,42	4,2	novembro 24, 2017, 8:14 PM
42	4893655420066	Awesome Rubber Wallet	Widget	Schamberger-Maggio	25,34	4	setembro 17, 2016, 6:21 PM
53	8825217022124	Mediocre Cotton Toucan	Doohickey	Jacobson-Daniel	29,52	4,4	novembro 4, 2016, 12:33 AM
56	8296484749050	Awesome Silk Car	Doohickey	Hackett-Reynolds	24,25	4,1	junho 28, 2017, 1:30 PM

Filtrando Registros

Operadores Lógicos

Com os **operadores lógicos**, é possível **combinar mais de uma condição** usando a cláusula **WHERE**. Estão disponíveis **três operadores lógicos** para a cláusula WHERE, sendo elas: **AND**, **OR**, **NOT**.

Função dos operadores lógicos:

AND	Retorna registros que atendem a todas as condições especificadas. Tanto a primeira quanto a segunda, ou as demais condições devem ser verdadeiras.
OR	Retorna registros que atendem a, pelo menos, uma das condições especificadas. A primeira, ou a segunda, ou as demais devem ser verdadeiras
NOT	Retorna todos os registros que não atendem a condição especificada.

Imagine que seja necessário retornar todos os registros da tabela **products** em que a avaliação (**rating**) seja **maior que 2 e menor que 4**. A query para essa consulta ficaria da seguinte forma:

```
select * from products where rating > 2 and rating < 4;
```

Filtrando Registros

Operadores de Comparação

Os **operadores de comparação** são utilizados em uma condição com a finalidade de **comparar dois valores** e **retornar verdadeiro ou falso**. Caso seja retornado **verdadeiro**, a **condição** comparada é **aceita**.

Função dos operadores de comparação:

<	Menor que: compara se o primeiro valor é menor que o segundo.
>	Maior que: compara se o primeiro valor é maior que o segundo.
<=	Menor ou igual: compara se o primeiro valor é menor ou igual ao segundo.
>=	Maior ou igual: compara se o primeiro valor é maior ou igual ao segundo.
=	Igual: compara se os dois valores são iguais.
!= ou **<>**	Diferente: compara se os dois valores são diferentes.

Filtrando Registros

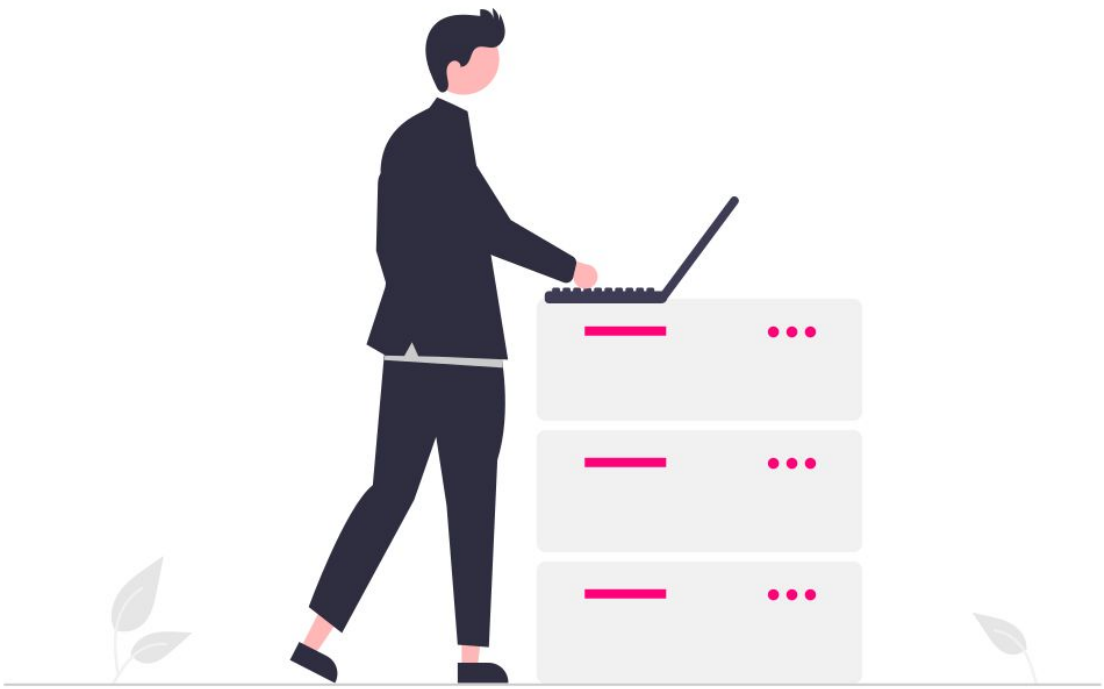
Predicados de Comparação

Dentre as formas de **comparar dois valores**, também podemos utilizar os chamados **predicados de comparação**. Os predicados têm a **mesma função** dos **operadores de comparação** e são representados por algumas **palavras reservadas**.

Função dos predicados de comparação:

BETWEEN	Compara se um valor está entre um intervalo de valores
NOT BETWEEN	Compara se um valor NÃo está entre um intervalo de valores
IS NULL	Verifica se o valor de um campo é nulo
IS NOT NULL	Verifica se o valor de um campo NÃo é nulo
LIKE	Permite comparar um valor ou uma parte dele
ILIKE	Permite comparar um valor ou uma parte dele sem diferenciar caracteres maiúsculos e minúsculos.

* Tanto o **LIKE** quanto o **ILIKE** aceitam coringas "**%**" e "**_**". Isso significa que o coringa **%** representa **qualquer correspondência até a sua posição** e o coringa **_** representa **apenas um caracter em sua posição**.



Ordenando Registros

Ordenando Registros

ORDER BY

Numa consulta SQL, os registros são **retornados aleatoriamente**, embora não pareça. Caso seja necessário **ordenar o retorno de uma consulta**, a cláusula **ORDER BY** pode ser usada. Essa cláusula permite **ordenar** o retorno de uma consulta por **um ou mais campos**, em ordem **crescente** ou **decrescente**.

Observe a instrução abaixo:

```
select id, name, email, birth_date, city
from people
order by name asc;
```

A query apresentada anteriormente, ao ser executada, retorna todos os registros da tabela **people** com os campos **id, name, email, birth_date e city**, ordenados pelo campo **name** em ordem **ascendente**.

ID	NAME	EMAIL	BIRTH_DATE	CITY
262	Aaron Hand	hand.aaron@hotmail.com	maio 17, 1964	Hardy
2210	Abbey Satterfield	satterfield-abbey@hotmail.com	junho 9, 1985	Wapato
624	Abbie Parisian	parisian.abbie@yahoo.com	julho 1, 1983	Pilot Hill
276	Abbie Ryan	abbie-ryan@hotmail.com	dezembro 30, 1971	Sultan

Ordenando Registros

ORDER BY coluna1 ASC, coluna2 DESC...

Caso seja necessário **ordenar os registros** por **mais de um campo**, a **prioridade de ordenação** será da **primeira coluna informada**, depois a **segunda** e assim sucessivamente.

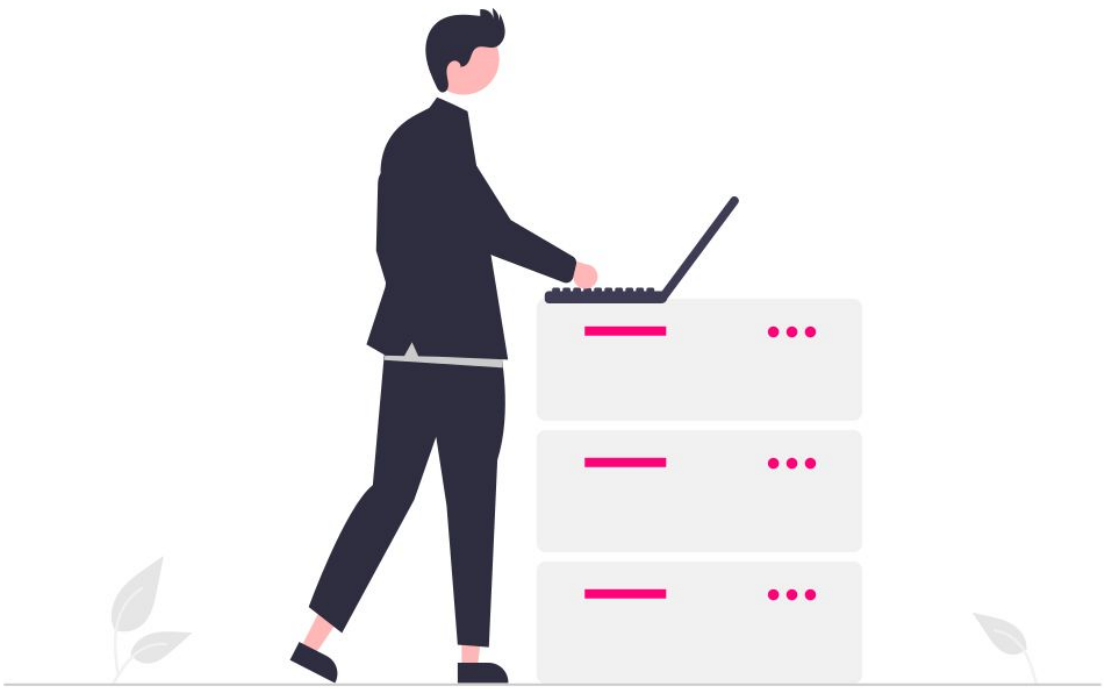
Digamos que seja necessário ordenar os registro da tabela **people** pelo campo **id** em **ordem ascendente** e **name** em **ordem descendente**. Temos a seguinte query:

```
select id, name, email, birth_date, city
from people
order by id asc, name desc;
```

A query apresentada anteriormente, ao ser executada, retorna o seguinte:

ID	NAME	EMAIL	BIRTH_DATE	CITY
1	Hudson Borer	borer-hudson@yahoo.com	dezembro 12, 1986	Wood River
2	Domenica Williamson	williamson-domenica@yahoo.com	junho 10, 1967	Searsboro
3	Lina Heaney	lina.heaney@yahoo.com	dezembro 18, 1961	Sandstone
4	Arnold Adams	adams.arnold@gmail.com	agosto 12, 1992	Rye
5	Dominique Leffler	leffler.dominique@hotmail.com	abril 20, 1974	Beaver Dams

SQL



Limitando e Omitindo Registros

Limitando e Omitindo Registros

LIMIT (FETCH)

Nem toda consulta precisa retornar todos os registros. Por isso, a cláusula **LIMIT** (originalmente o SQL dá o nome de FETCH) existe e tem a função de **limitar a quantidade de registros** que são retornados na **consulta**, partindo do primeiro até o limite de registros informado. O limite precisa ser um **valor positivo** partindo do **número 1**.

Digamos que precise retornar os 10 primeiros registros da tabela **orders**, observe a instrução abaixo:

```
select * from orders limit 10;
```

A query apresentada anteriormente, ao ser executada, retorna o seguinte:

ID	USER_ID	PRODUCT_ID	SUBTOTAL	TAX	TOTAL	DISCOUNT	CREATED_AT	QUANTITY
1	1	14	37,65	2,07	39,72		fevereiro 11, 2019, 9:40 PM	2
2	1	123	110,93	6,1	117,03		maio 15, 2018, 8:04 AM	3
3	1	105	52,72	2,9	49,21	6,42	dezembro 6, 2019, 10:22 PM	2
4	1	94	109,22	6,01	115,23		agosto 22, 2019, 4:30 PM	6
5	1	132	127,88	7,03	134,91		outubro 10, 2018, 3:34 AM	5
6	1	60	29,8	1,64	31,44		novembro 6, 2019, 4:38 PM	3
7	1	55	95,77	5,27	101,04		setembro 11, 2018, 11:22 AM	5
8	1	65	68,23	3,75	63,32	8,65	junho 17, 2019, 2:37 AM	7
9	1	184	77,4	4,26	78,06	3,59	maio 3, 2017, 4:00 PM	3
10	1	6	97,44	5,36	102,8		janeiro 17, 2020, 1:44 AM	2

Limitando e Omitindo Registros

OFFSET

A cláusula **OFFSET** **omite os resultados**, partindo do **primeiro registro**. Isso significa que a consulta retornará todos os registros solicitados **sem os registros omitidos**. O **OFFSET** recebe um valor **inteiro positivo**, **partindo do 0**, embora um **OFFSET 0** não tenha qualquer impacto na query. Geralmente, essa cláusula é usada em conjunto com a cláusula **LIMIT**.

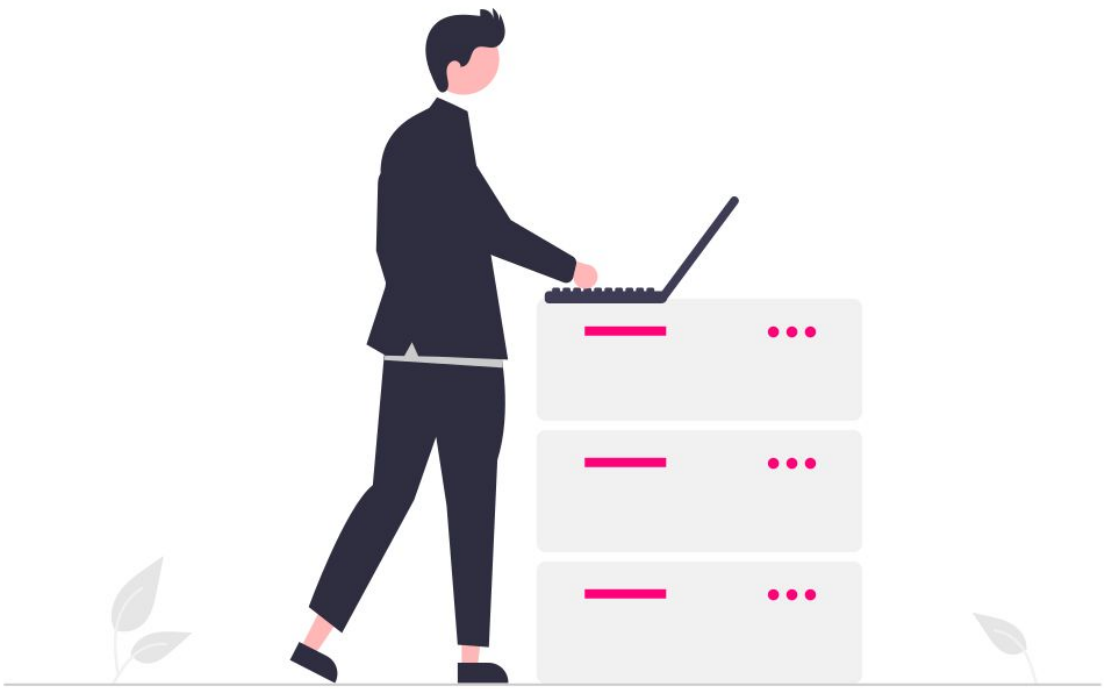
Digamos que precise retornar os 10 primeiros registros da tabela **orders**, pulando os 3 primeiros registros, observe a instrução abaixo:

```
select * from orders limit 10 offset 3;
```

A query apresentada anteriormente, ao ser executada, retorna o seguinte:

ID	USER_ID	PRODUCT_ID	SUBTOTAL	TAX	TOTAL	DISCOUNT	CREATED_AT	QUANTITY
4	1	94	109,22	6,01	115,23		agosto 22, 2019, 4:30 PM	6
5	1	132	127,88	7,03	134,91		outubro 10, 2018, 3:34 AM	5
6	1	60	29,8	1,64	31,44		novembro 6, 2019, 4:38 PM	3
7	1	55	95,77	5,27	101,04		setembro 11, 2018, 11:22 AM	5
8	1	65	68,23	3,75	63,32	8,65	junho 17, 2019, 2:37 AM	7
9	1	184	77,4	4,26	78,06	3,59	maio 3, 2017, 4:00 PM	3
10	1	6	97,44	5,36	102,8		janeiro 17, 2020, 1:44 AM	2
11	1	76	63,82	3,51	67,33		julho 22, 2018, 8:31 PM	6
12	3	7	148,23	10,19	158,42		junho 26, 2018, 11:21 PM	7
13	3	70	57,49	3,95	59,33	2,12	abril 6, 2019, 1:04 AM	2

SQL



Consultando Registros Únicos

Consultando Registros Únicos

SELECT DISTINCT

O **SELECT** pode ser usado em conjunto com a cláusula **DISTINCT** com a finalidade de retornar os **registros únicos** de uma tabela, **eliminando duplicidade**. Geralmente, são informados **um ou mais campos combinados** e cada registro da combinação é **único**.

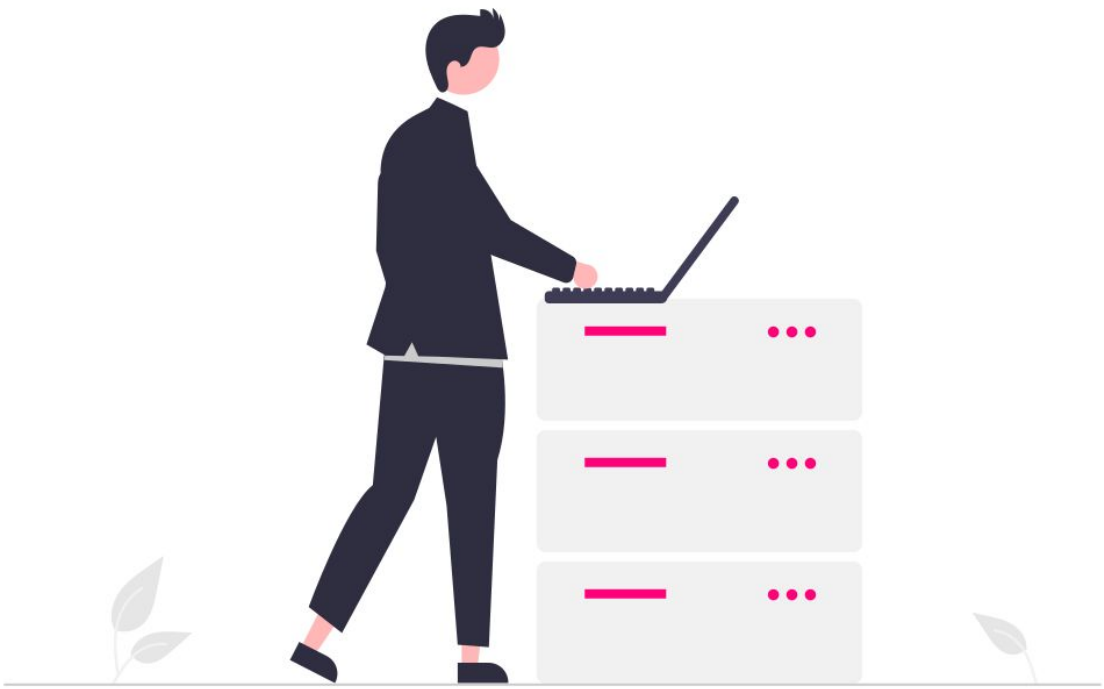
Digamos que precise retornar todas as cidades existentes da tabela **people**. Observe a query abaixo:

```
select distinct city from people;
```

A query apresentada anteriormente, ao ser executada, retorna o seguinte:

CITY
Lyle
Riverton
Hillsdale
San Marcos
Hill City
Newton Grove
Greentown
Battle Lake
Cornelius

SQL



Expressões e Funções

Expressões e Funções

Operadores Matemáticos

Os **operadores matemáticos** possibilitam criar **expressões** matemáticas em uma query para **manipular e retornar** valores entre os **campos de uma tabela**.

Função dos operadores matemáticos:

+	Soma valores
-	Subtração de valores
*****	Multiplicação de valores
/	Divisão de valores
%	Resto da divisão entre dois valores
^	Potência

Imagine que precise retornar todos os registros da tabela **products**, onde, aplicando o desconto de 10% no campo **price** não fique **menor que 80**. A query ficaria assim:

```
select * from products where price - (price * 0.1) >= 80;
```

Expressões e Funções

Funções Matemáticas

Além dos **operadores matemáticos** temos as **funções matemáticas** que podem ser usadas em **um campo** para **manipulação do valor** específico.

Funções matemáticas:

abs	Obtém o valor absoluto de um número
ceil	Arredonda para o próximo inteiro superior
floor	Arredonda para o próximo inteiro inferior
round	Arredonda para o inteiro mais próximo
sqrt	Obtém a raiz quadrada

Imagine que a avaliação do produto seja um número inteiro e na consulta não pode ser retornado fracionado. Nesse caso, podemos arredondar o valor da coluna rating de acordo com a regra da consulta usando uma das funções acima. Exemplo:

```
select *, round(rating) from products;
```

Expressões e Funções

Operadores e Funções com Strings

É possível **manipular os valores dos campos** que armazenam **caracteres**, também chamados de **"strings"**, usando **operadores e/ou funções**. Algumas funções podem **variar** de acordo com o **banco de dados**, mas a maioria delas são comuns.

Funções e operadores para manipulação de strings:

||	Concatena strings
concat	Função para concatenação
char_length	Retorna a quantidade de caracteres
lower	Converte todos os caracteres para minúsculos
upper	Converte todos os caracteres para maiúsculos

Imagine que o nome de todas as pessoas precisasse ser retornado em uma consulta, em maiúsculo. Nesse caso, podemos usar a função upper. Exemplo:

```
select id, upper(name) as name, email from people;
```

* O alias "**as**" é usado para criar um **apelido personalizado** para o nome do campo.

Expressões e Funções

Conversão de Tipos

É possível **converter um tipo de dado** de um **campo** para qualquer outro tipo, **em uma consulta**, usando o operador **"::"** ou usando a função **cast**, caso seja implementada pelo banco de dados em uso. A **conversão só é válida no momento da consulta** e **não altera o tipo de dado nativo da coluna**.

Visivelmente não muda nada, mas o tipo de dado daquele campo na consulta, passa a ser do tipo convertido. Vejamos alguns exemplos de conversão de tipos.

Convertendo um número para um texto.

```
select *, rating::text from products;
```

Convertendo um número para um texto usando a função **cast**.

```
select *, cast(rating as text) from products;
```

Expressões e Funções

Funções com Datas e Intervalos

Frequentemente estamos trabalhando com **datas no banco de dados**. Por isso, também podemos encontrar **funções para manipular campos que armazenam datas**. As datas possuem **tipos** (formatos) específicos e as **funções manipulam esses tipos**.

Tipos de datas:

timestamp	Data e hora sem fuso horário. **Ex.: 2021-05-20 12:30:45**
timestamptz	Data e hora com fuso horário. **Ex.: 2021-05-20 12:30:45-03**
date	Data sem hora do dia. **Ex.: 2021-05-20**
time	Hora do dia sem data. **Ex.: 12:30:45**
timetz	Hora do dia sem data + fuso. **Ex.: 12:30:45-03**

Funções com datas e intervalos:

now	Obtém a data atual do banco de dados
extract	Extrai parte de uma data ou hora específica
at time zone	Converte a data em um fuso horário para outro

Expressões e Funções

Funções com Datas e Intervalos

Vejamos alguns exemplos de manipulação de datas.

Obtendo a data atual do banco de dados:

```
select now();
```

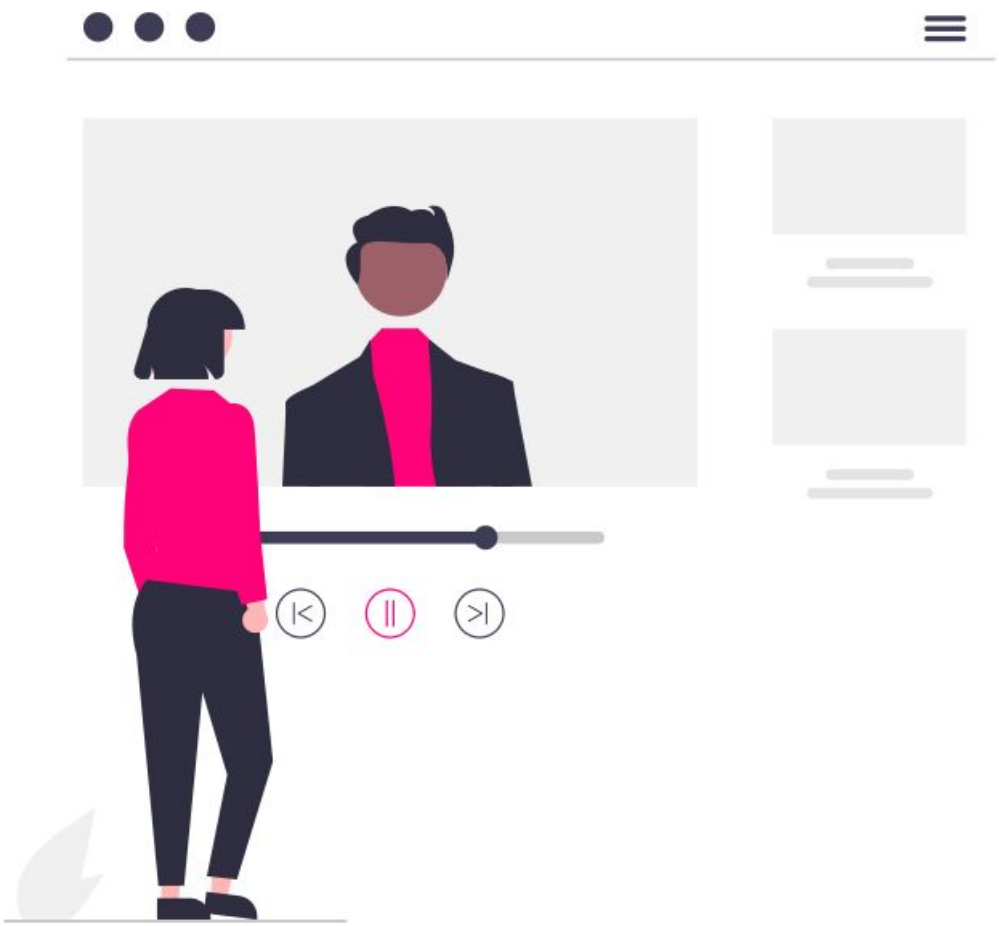
Extraindo o ano da data de criação de cada registro na tabela produto.

```
select *, extract(year from created_at) from products;
```

Obtendo a data no time zone "America/Sao_Paulo"

```
select *, created_at at time zone "America/Sao_Paulo"
from products;
```

Guido Cerqueira
Professor Cubos Academy



cubos //academy//

Guido Cerqueira
Professor Cubos Academy

cubos
// academy //

www.cubos.academy