**Criando o nosso banco de dados**

Durante todo o curso usaremos o terminal do MySQL. Apesar dele não ser a melhor interface gráfica, ele é igual para todas as plataformas, e não muda de versão para versão. Se você já conhecer alguma, não há problema de fazer uso dela.

Abra o terminal do seu sistema operacional. No Windows, digite cmd no Executar. No Mac e Linux, abra o terminal. Nele, vamos entrar no MySQL:

mysql -uroot -p

Esse é o comando para nos conectarmos no MySQL que está instalado na máquina. O -u indica o usuário root, e o -p é porque digitaremos a senha. Como não há senha, pressione enter.

Vamos agora criar um banco de dados. O create database cria, e o use diz ao MySQL que é o banco que queremos usar.

create database controle\_compras;

use controle\_compras;

para visualizar as tabelas dentro do banco de dados – depois que vc deu use database

show tables;

Escreva o comando desc para visualizar os atributos de sua tabela já criada

desc Compras;

Agora, vamos criar nossa primeira tabela

create table COMPRAS (id int auto\_increment primary key, valor double, data date, observacoes varchar(255), recebido boolean);

**Consultando dados**

Queremos inserir na tabela COMPRAS, então começamos a instrução:

INSERT INTO COMPRAS

Em seguida, passamos os valores que queremos inserir. Por exemplo

INSERT INTO COMPRAS (VALOR, DATA, OBSERVACOES, RECEBIDO) VALUES (100.0, '2007-05-12', 'COMPRAS DE MAIO', 1)

Pronto. Com essa compra já inserida nesse banco de dados, vamos começar a consultá-las. Se quiséssemos, por exemplo, ver todas as compras já cadastradas na minha lista, faríamos:

SELECT \* FROM COMPRAS

Se quiséssemos selecionar apenas valor e data dessa tabela, faríamos:

SELECT VALOR, DATA FROM COMPRAS

Podemos inclusive criar novas colunas a partir das já existentes. Por exemplo, caso quiséssemos exibir, além de valor e data, o triplo de cada valor, poderíamos fazer:

SELECT VALOR, VALOR \* 3, DATA FROM COMPRAS

Podemos até dar um nome para essa nova coluna, por exemplo, "triplo", usando a instrução AS:

SELECT VALOR, VALOR \* 3 AS TRIPLO, DATA FROM COMPRAS

Geralmente aplicamos filtros sobre esses resultados. Vamos supor que precisássemos saber todas as compras com valor superior a 1.000,00 reais. Poderíamos deixar esse filtro claro na instrução SQL:

SELECT \* FROM COMPRAS WHERE VALOR > 1000

Podemos montar filtros ainda mais complexos. Por exemplo, caso precisássemos de todas as compras com valor superior a 1.000,00 reais e inferior a 3.000,00, poderíamos fazer:

SELECT \* FROM COMPRAS WHERE VALOR > 1000 AND VALOR < 3000

filtrar todas as compras cujo valor é superior a 1.000,00 reais e inferiores a 3.000,00 ou a data seja inferior a 12/02/2010:

SELECT \* FROM COMPRAS WHERE (VALOR > 1000 AND VALOR < 3000) OR (DATA < '2010-02-12')

Além de maior e menor, a SQL suporta outros tipos de consultas, como por exemplo:

Maior-ou-igual:

SELECT \* FROM COMPRAS WHERE VALOR >= 1000

Menor-ou-igual:

SELECT \* FROM COMPRAS WHERE VALOR <= 1000

Diferente:

SELECT \* FROM COMPRAS WHERE VALOR <> 1000

Podemos filtrar também por textos. Se quiséssemos, por exemplo, buscar todas as compras cuja observação seja igual a "COMPRAS DE JANEIRO", podemos fazer:

SELECT \* FROM COMPRAS WHERE OBSERVACOES = 'COMPRAS DE JANEIRO'

caso quiséssemos todas as compras cuja observação começasse com o texto "COMPRAS", independentemente do que viesse a seguir, deveríamos fazer:

SELECT \* FROM COMPRAS WHERE OBSERVACOES LIKE 'COMPRAS%'

O caractere % funciona como um coringa, ou seja, não importa o que houver dali pra frente.

caso quiséssemos todas as compras cuja observação contenha a palavra "COMPRA" em qualquer lugar, faríamos:

SELECT \* FROM COMPRAS WHERE OBSERVACOES LIKE '%COMPRAS%'

A instrução SELECT é poderosíssima. Boa sorte nos exercícios!

**02 Atualizando e excluindo dados**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/introducao-a-banco-de-dados-e-sql/task/5667/next)

desejamos selecionar apenas as que possuem valor entre R$200,00 e R$700,00. Para isso usaremos a cláusula WHERE:

SELECT \* FROM COMPRAS WHERE valor >= 200 AND valor <= 700

O operador BETWEEN serve para filtrar registros de acordo com um intervalo de valores. É possível filtrar apenas as compras realizadas entre 05/01/2010 e 25/06/2010.

SELECT \* FROM COMPRAS WHERE DATA BETWEEN '2010-01-05' AND '2010-06-25'

Mas o que precisamos fazer é alterar os registros:

"Atualize as compras, alterando a observação para 'compra emergencial'"

Podemos fazer um UPDATE como o a seguir:

UPDATE COMPRAS SET OBSERVACOES = 'compra emergencial'

"Atualize as compras, alterando a observação para 'uma compra comum', somente se a data não estiver entre os dias 05-01 e 25-06."

Traduzindo para sql:

UPDATE COMPRAS SET OBSERVACOES = 'uma compra comum' WHERE DATA NOT BETWEEN '2010-01-05' AND '2010-06-25'

O operador IN foi usado para criar um conjunto de valores que usamos na comparação. O trecho WHERE DATA IN ('2010-12-25', '2010-10-12', '2010-06-12') diz que queremos todos os registros com qualquer uma das datas que esteja entre os parênteses,

UPDATE COMPRAS SET OBSERVACOES = 'datas festivas' WHERE DATA IN('2010-12-25', '2010-10-12', '2010-06-12')

"Excluir todas as compras efetuadas antes de 2009":

DELETE FROM COMPRAS WHERE DATA < '2009-01-01'

------------------------------------------------------

# 02Agrupando dados e fazendo consultas mais inteligentes

Vamos supor que precisamos calcular a soma de todas as compras realizadas até hoje na coluna VALOR. Usando a função SUM, temos o resultado esperado:

SELECT SUM(VALOR) FROM COMPRAS

Por ser um comando SELECT, podemos colocar condições, por exemplo, calcular a soma de todas as compras feitas antes de 01/01/2010:

SELECT SUM(VALOR) FROM COMPRAS WHERE DATA < '2010-01-01'

Podemos calcular informações ainda mais interessantes, tal como a média das compras feitas antes de 01/01/2010. Basta usarmos a função AVG:

SELECT AVG(VALOR) FROM COMPRAS WHERE DATA < '2010-01-01'

Podemos inclusive juntar as duas, dando nomes mais bonitos para as colunas geradas, usando o comando AS, visto nas aulas anteriores:

SELECT AVG(VALOR) AS MEDIA, SUM(VALOR) AS SOMA FROM COMPRAS WHERE DATA < '2010-01-01'

Podemos contar o número de compras da mesma forma, usando a função COUNT:

SELECT COUNT(VALOR) FROM COMPRAS WHERE DATA < '2010-01-01'

Percebemos até então que o comando SELECT unido às funções de soma, média e contagem pode gerar dados muito úteis. Mas e se quisermos a soma de todas as compras recebidas e não recebidas? Podemos fazer duas consultas, por exemplo:

SELECT SUM(VALOR) FROM COMPRAS WHERE RECEBIDO = 1;

SELECT SUM(VALOR) FROM COMPRAS WHERE RECEBIDO = 0;

E se quisermos todo esse resultado em apenas uma única consulta? Afinal, executar uma-a-uma manualmente pode gerar muito trabalho - imagine se quiséssemos a soma das compras por forma de pagamento, seriam 3 consultas diferentes; uma para BOLETO, uma para CARTAO, uma para DINHEIRO.

Para isso, precisamos informar ao MySQL que queremos agrupar os dados. No nosso caso, queremos agrupar compras de acordo com o valor da coluna RECEBIDO. Usamos então o comando GROUP BY, junto ao resto da SQL. Veja o exemplo:

SELECT SUM(VALOR) FROM COMPRAS GROUP BY RECEBIDO

Veja que agora ele agrupou e exibiu dois resultados, muito provavelmente a soma de todas as compras na qual recebido vale 1, e as compras na qual recebido vale 0. O problema é que não sabemos qual valor corresponde a cada uma das condições de RECEBIDO. Para isso, podemos exibir a coluna RECEBIDO na consulta também:

SELECT RECEBIDO, SUM(VALOR) FROM COMPRAS GROUP BY RECEBIDO

Agora sabemos a soma para cada grupo! Podemos juntar também os valores desse agrupamento para exibir a soma e a quantidade de itens em cada grupo, por exemplo:

SELECT RECEBIDO, SUM(VALOR) AS SOMA, COUNT(VALOR) AS TOTAL FROM COMPRAS GROUP BY RECEBIDO

Podemos refinar ainda mais esse resultado, ordenando, por exemplo, pelo grupo na qual a soma seja maior (ou seja, de maneira decrescente). Para isso, usamos o comando ORDER BY, informando o campo que queremos ordenar no final da nossa consulta SQL:

SELECT RECEBIDO, SUM(VALOR) AS SOMA, COUNT(VALOR) AS TOTAL FROM COMPRAS GROUP BY RECEBIDO ORDER BY SOMA DESC

Repare a palavra DESC. Ela indica que queremos ordenar de maneira decrescente. Para ordenar de maneira crescente, fazemos uso da palavra ASC:

SELECT RECEBIDO, SUM(VALOR) AS SOMA, COUNT(VALOR) AS TOTAL FROM COMPRAS GROUP BY RECEBIDO ORDER BY SOMA ASC

Consultas agrupadas são um recurso muito interessante e permite-nos extrair muitas informações dos nossos dados!

# -----------------------------------------

# Juntando dados de várias tabelas

Vamos primeiro criar uma tabela para representar todas as pessoas que já fizeram uma compra:

CREATE TABLE COMPRADORES (

ID INT NOT NULL AUTO\_INCREMENT,

NOME VARCHAR(100) NOT NULL,

ENDERECO VARCHAR(100) NOT NULL,

TELEFONE VARCHAR(20) NOT NULL,

PRIMARY KEY(ID)

)

Precisamos dizer ao MySQL que queremos juntar esses dados, e que a coluna que faz essa junção é justamente a coluna COMPRADOR\_ID da tabela COMPRAS com a coluna ID da tabela COMPRADORES. Basta dizermos que essa é a coluna que faz a junção ou, em inglês, o JOIN:

SELECT \* FROM COMPRAS JOIN COMPRADORES ON COMPRAS.COMPRADOR\_ID = COMPRADORES.ID

Agora sim, o MySQL tem informações suficientes para juntar os dados das duas

**Chave estrangeira**

Vamos fazer alteração na tabela, e dizer que a coluna COMPRADOR\_ID é uma FK:

ALTER TABLE COMPRAS ADD FOREIGN KEY (COMPRADOR\_ID) REFERENCES COMPRADORES(ID)

Veja que indicamos nessa instrução SQL o nome da coluna que é uma FK (no caso, COMPRADOR\_ID) e dissemos que ela aponta para o campo ID da tabela COMPRADORES.

Os JOINs nos permitem juntar tabelas que se relacionam através das FKs. É um recurso poderoso que você deve praticar!