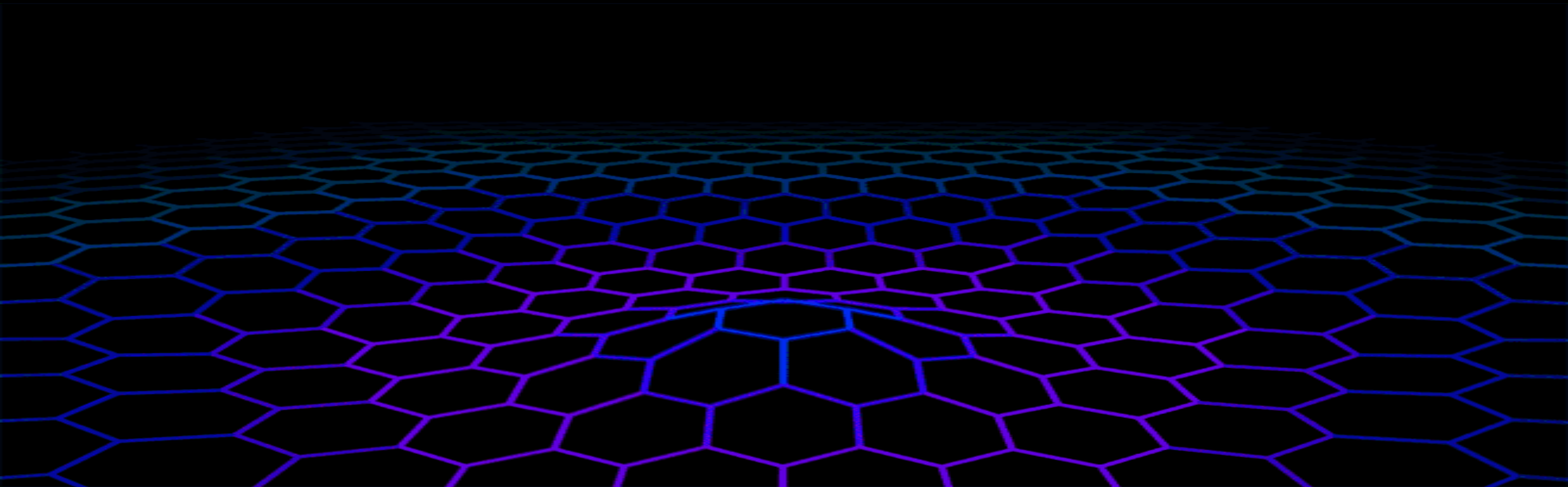
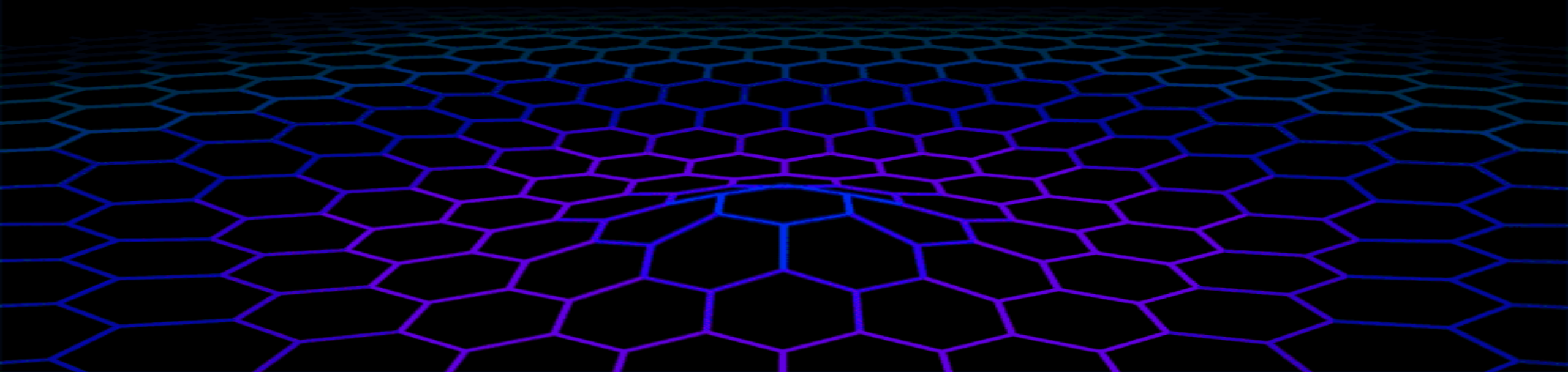
Banco de Dados 2



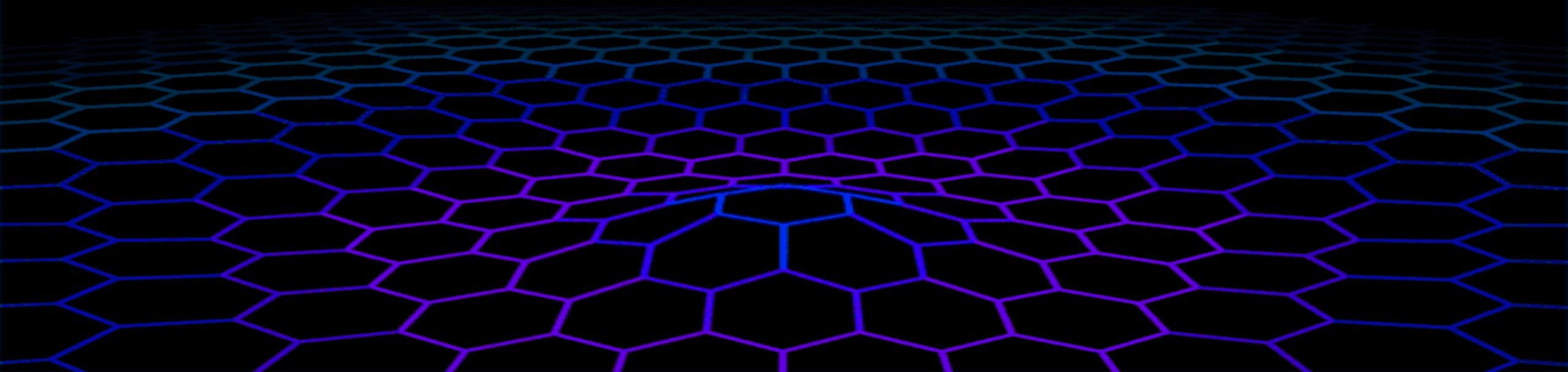
Faculdades senac

Prepare-se!



Faculdades senac

Prepare-se!



Hoje tem!

Dicas + Case + Pivotamento


FICA!
&DICA!

#1

#1 DELETE COM WHERE

#1 DELETE COM WHERE

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id       INT(8) NOT NULL AUTO_INCREMENT,
     nome     VARCHAR(100),
     valor    DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo    BOOLEAN,
     PRIMARY KEY (id)
  );
```

#1 DELETE COM WHERE

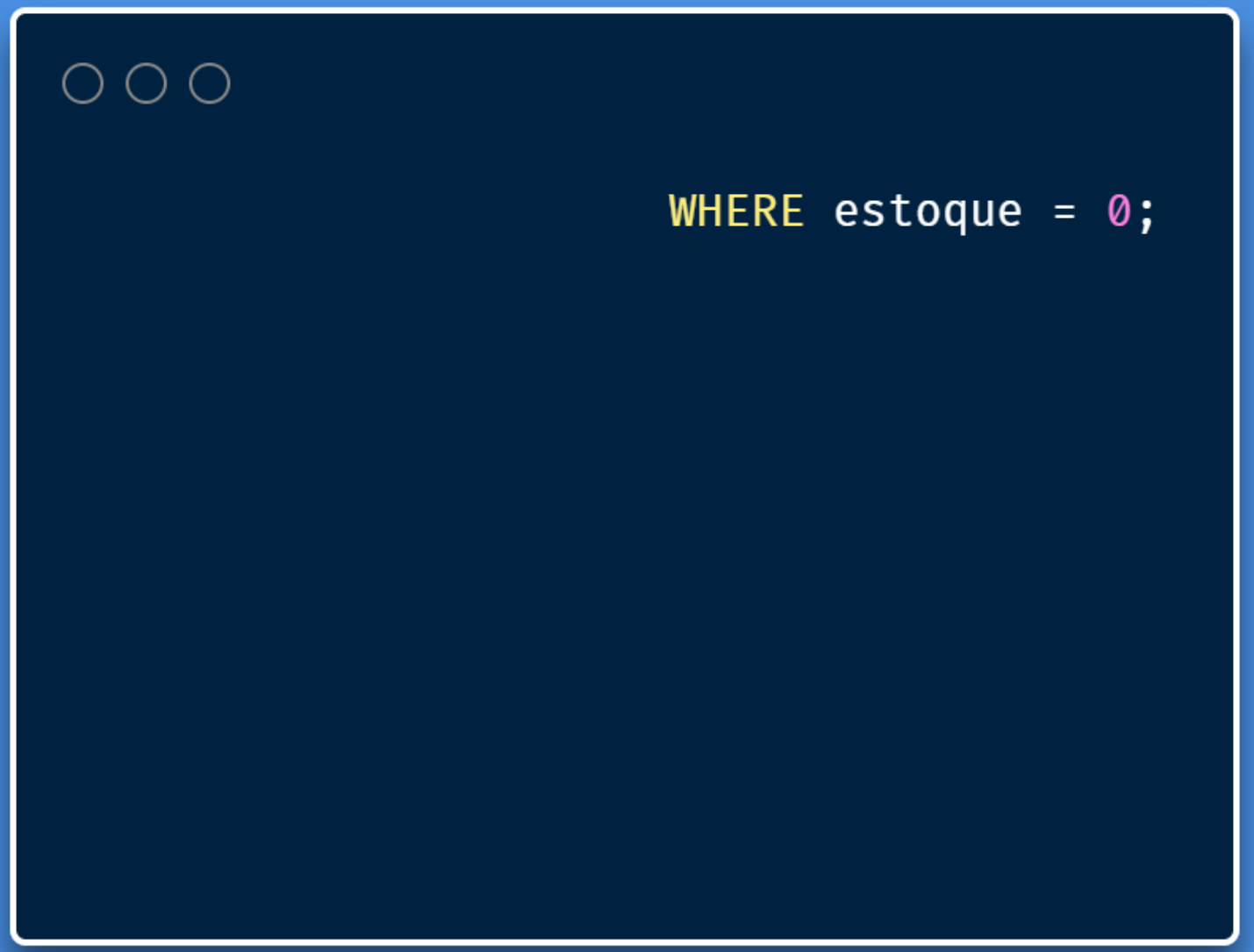
```sql
INSERT INTO produto VALUES (1,"Livro S", 99.99, 12, TRUE);
INSERT INTO produto VALUES (2,"Livro E", 89.99, 0, TRUE);
INSERT INTO produto VALUES (3,"Livro N", 49.99, 3, TRUE);
INSERT INTO produto VALUES (4,"Livro A", 56.38, 2, TRUE);
INSERT INTO produto VALUES (5,"Livro C", 32.41, 0, TRUE);
```

#1 DELETE COM WHERE

Sempre que escrevemos um DELETE no console do banco de dados vale a pena começar com o WHERE.

A dica também serve para o comando UPDATE.

#1 Iniciar pelo WHERE



#1 Fazer um SELECT

```sql
SELECT * FROM produto WHERE estoque = 0;
```

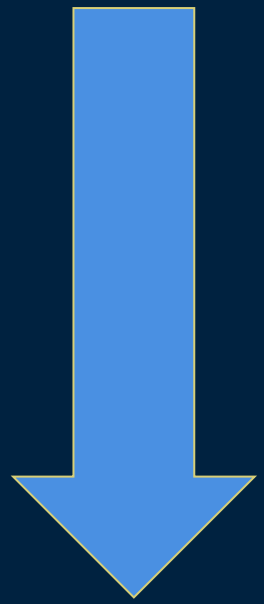
#1 Conferir o resultado

```
SELECT * FROM produto WHERE estoque = 0;

+----+-----------+-------+----------+-------+
| id | nome      | valor | estoque | ativo |
+----+-----------+-------+----------+-------+
|  2 | Livro E | 89.99 |     0.00 |     1 |
|  5 | Livro C | 32.41 |     0.00 |     1 |
+----+-----------+-------+----------+-------+
2 rows in set (0.00 sec)
```

#1 Substituir o SELECT por DELETE

```sql
SELECT * FROM produto WHERE estoque = 0;
```



```sql
DELETE FROM produto WHERE estoque = 0;
```

Assim você evita que um [ENTER] acidental apague sua tabela toda.

#2

#2 Promoção com Livro por 2 Reais? Ou ainda -2 Reais?

Alguns campos devem ser validados declarativamente, como o preço de um produto;

#2 Promoção com Livro por 2 Reais? Ou ainda -2 Reais?

Alguns campos devem ser validados declarativamente, como o preço de um produto;

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id     INT(8) NOT NULL AUTO_INCREMENT,
     nome   VARCHAR(100),
     valor DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo BOOLEAN,
     PRIMARY KEY (id)
  );
```

#2 Promoção com Livro por 2 Reais? Ou ainda -2 Reais?

Alguns campos devem ser validados declarativamente, como o preço de um produto;

```
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id     INT(8) NOT NULL AUTO_INCREMENT,
     nome   VARCHAR(100),
     valor DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo BOOLEAN,
     PRIMARY KEY (id)
  );

INSERT INTO produto VALUES (8,"Livro X",-2.00, 12, TRUE);
-- Query OK, 1 row affected (0.00 sec)
```

#2 Promoção com Livro por 2 Reais? Ou ainda -2 Reais?

Alguns campos devem ser validados declarativamente, como o preço de um produto;

Inserindo linha com preço negativo :(
Problema! Eu queria integridade dos meus dados.
Queria que ele checasse por valores inválidos.

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id     INT(8) NOT NULL AUTO_INCREMENT,
     nome   VARCHAR(100),
     valor DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo BOOLEAN,
     PRIMARY KEY (id)
  );

INSERT INTO produto VALUES (8,"Livro X",-2.00, 12, TRUE);
-- Query OK, 1 row affected (0.00 sec)
```

#2 Manda o banco checar o valor

#2 Manda o banco checar o valor

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id      INT(8) NOT NULL AUTO_INCREMENT,
     nome    VARCHAR(100),
     valor   DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo   BOOLEAN,
     PRIMARY KEY (id),
     CHECK(valor > 0)
  );
```

#2 Manda o banco checar o valor

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id      INT(8) NOT NULL AUTO_INCREMENT,
     nome    VARCHAR(100),
     valor   DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo   BOOLEAN,
     PRIMARY KEY (id),
     CHECK(valor > 0)
  );

INSERT INTO produto VALUES (8,"Livro X",-2.00, 12, TRUE);
/* ERROR 3819 (HY000):
Check constraint 'produto_chk_1' is violated.*/
```

#2 Manda o banco checar o valor

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id      INT(8) NOT NULL AUTO_INCREMENT,
     nome    VARCHAR(100),
     valor   DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo   BOOLEAN,
     PRIMARY KEY (id),
     CHECK(valor > 0)
  );

INSERT INTO produto VALUES (8,"Livro X",-2.00, 12, TRUE);
/* ERROR 3819 (HY000):
Check constraint 'produto_chk_1' is violated.*/
```

Ainda bem que o MySQL e o MariaDB já suportam **CHECK** (da mesma forma que Oracle, PostgreSQL, SQL Server, ...)

#3

#3

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id       INT(8) NOT NULL AUTO_INCREMENT,
     nome     VARCHAR(100),
     valor    DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo    BOOLEAN,
     PRIMARY KEY (id),
     CHECK(valor > 0)
  );
```

#3

```sql
INSERT INTO produto VALUES (10,"Livro A",1.09, 10, TRUE);
INSERT INTO produto VALUES (11,"Livro B",2000.56, 12, TRUE);
INSERT INTO produto VALUES (12,"Livro C",3000.99, 2, FALSE);
INSERT INTO produto VALUES (13,"Livro D",45.99, 9, TRUE);
INSERT INTO produto VALUES (14,"Livro E",2.99, 11, FALSE);
INSERT INTO produto VALUES (15,"Livro F",80.19, 10, FALSE);
INSERT INTO produto VALUES (16,"Livro G",39.21, 10, TRUE);
INSERT INTO produto VALUES (17,"Livro H",68.23, 8, TRUE);
INSERT INTO produto VALUES (19,"Livro I",2.00, 3, TRUE);
INSERT INTO produto VALUES (20,"Livro J",2500.38, 4, FALSE);
```

#3

Desafio:

Gostaria de trazer os livros ativos cujos preços tem algo de estranho: são muito baratos (< 10) OU muito caros (> 1000)

Qual o problema da query?

```sql
SELECT *
FROM   produto
WHERE  ativo = true
   AND valor < 10
    OR valor > 1000;
```

#3

Desafio:

Gostaria de trazer os livros ativos cujos preços tem algo de estranho: são muito baratos (< 10) OU muito caros (> 1000)

Qual o problema da query?

Query bonita escrita, executada, resultado "errado".
Quem nunca teve uma query respondendo um monte de coisa que não foi pedido?

```
SELECT *
FROM   produto
WHERE  ativo = true
   AND valor < 10
    OR valor > 1000;
```

```
+----+---------+---------+----------+-------+
| id | nome    | valor   | estoque | ativo |
+----+---------+---------+----------+-------+
| 10 | Livro A |    1.09 |   10.00 |     1 |
| 11 | Livro B | 2000.56 |   12.00 |     1 |
| 12 | Livro C | 3000.99 |    2.00 |     0 |
| 19 | Livro I |    2.00 |    3.00 |     1 |
| 20 | Livro J | 2500.38 |    4.00 |     0 |
+----+---------+---------+----------+-------+
5 rows in set (0.00 sec)
```

#3

Desafio:

Gostaria de trazer os livros ativos cujos preços tem algo de estranho: são muito baratos (< 10) OU muito caros (> 1000)

Qual o problema da query?

Query bonita escrita, executada, resultado "errado".
Quem nunca teve uma query respondendo um monte de coisa que não foi pedido?

Problema: O MySQL poderá retornar livros inativos.

```
SELECT *
FROM   produto
WHERE  ativo = true
   AND valor < 10
    OR valor > 1000;
```

```
+----+---------+---------+----------+-------+
| id | nome    | valor   | estoque | ativo |
+----+---------+---------+----------+-------+
| 10 | Livro A |    1.09 |   10.00 |     1 |
| 11 | Livro B | 2000.56 |   12.00 |     1 |
| 12 | Livro C | 3000.99 |    2.00 |     0 |
| 19 | Livro I |    2.00 |    3.00 |     1 |
| 20 | Livro J | 2500.38 |    4.00 |     0 |
+----+---------+---------+----------+-------+
5 rows in set (0.00 sec)
```

#3 Parenteses

```sql
SELECT *
FROM   produto
WHERE  ativo = true
   AND (valor < 10
    OR valor > 1000);
```

#3 Parenteses

```
SELECT *
FROM   produto
WHERE  ativo = true
   AND (valor < 10
    OR valor > 1000);
```

```
+----+----------+----------+----------+--------+
| id | nome     | valor    | estoque  | ativo  |
+----+----------+----------+----------+--------+
| 10 | Livro A  |     1.09 |    10.00 |      1 |
| 11 | Livro B  | 2000.56  |    12.00 |      1 |
| 19 | Livro I  |     2.00 |     3.00 |      1 |
+----+----------+----------+----------+--------+
3 rows in set (0.00 sec)
```

A sacada? Sempre que escrever uma regra que envolva um OR, utilize parênteses.

#4

#4 Que tal criar uma tabela a partir de outra?

#4 Que tal criar uma tabela a partir de outra?

```
CREATE TABLE produtoClone LIKE produto;
```

#4 Bora fazer um Chupacabra

```sql
CREATE TABLE produtoClone LIKE produto;

INSERT INTO produtoClone
SELECT *
FROM produto;
```

#4

Se você quiser a tabela exista somente durante a sua conexão atual crie ela de maneira **temporária**:

#5

#5

```sql
DROP TABLE IF EXISTS produto;
CREATE TABLE IF NOT EXISTS produto
  (
     id       INT(8) NOT NULL AUTO_INCREMENT,
     nome     VARCHAR(100),
     valor    DECIMAL(10, 2),
     estoque DECIMAL(10, 2),
     ativo    BOOLEAN,
     PRIMARY KEY (id),
     CHECK(valor > 0)
  );
```

#5

```sql
INSERT INTO produto VALUES (21,NULL,38.42, 4, TRUE);
INSERT INTO produto VALUES (22,'',37.42, 8, TRUE);
INSERT INTO produto VALUES (23,"Livro XA",36.42, 10, TRUE);
INSERT INTO produto VALUES (24,"",35.42, 11, TRUE);
INSERT INTO produto VALUES (25,'Livro XB',34.42, 15, TRUE);
INSERT INTO produto VALUES (26,'Livro XC',33.42, 2, TRUE);
INSERT INTO produto VALUES (27,'Livro XD',32.42, 8, TRUE);
INSERT INTO produto VALUES (28,NULL,31.42, 4, TRUE);
INSERT INTO produto VALUES (29,'Livro XE',30.42, 3, TRUE);
INSERT INTO produto VALUES (30,'Livro XF',9.32, 2, TRUE);
INSERT INTO produto (valor, estoque, ativo) VALUES (9.32, 2, TRUE);
INSERT INTO produto (valor, estoque, ativo) VALUES (8.45, 3, TRUE);
```

#5

```sql
SELECT COUNT(nome) FROM produto;
```

#5

```
SELECT COUNT(nome) FROM produto;
+---------------+
| COUNT(nome) |
+---------------+
|           8 |
+---------------+
1 row in set (0.00 sec)
```

#5

```sql
SELECT COUNT(id) FROM produto;
```

#5

```
SELECT COUNT(id) FROM produto;
+-------------+
| COUNT(id) |
+-------------+
|         12 |
+-------------+
1 row in set (0.00 sec)
```

#5

Qual resultado é o correto?

8 ou **12**?

```
SELECT COUNT(nome) FROM produto;
+---------------+
| COUNT(nome) |
+---------------+
|           8 |
+---------------+
1 row in set (0.00 sec)

SELECT COUNT(id) FROM produto;
+-------------+
| COUNT(id) |
+-------------+
|        12 |
+-------------+
1 row in set (0.00 sec)
```

#5

Qual o resultado?

```sql
SELECT id, nome
FROM produto
WHERE nome IS NULL;
```

#5

Qual o resultado?

```
SELECT id, nome
FROM produto
WHERE nome IS NULL;
+----+------+
| id | nome |
+----+------+
| 21 | NULL |
| 28 | NULL |
| 31 | NULL |
| 32 | NULL |
+----+------+
4 rows in set (0.00 sec)
```

#5

Se você quer contar valores **NÃO NULOS**, use o **`COUNT(campo)`**

Se você quer contar **todos**, use **`COUNT(*)`**.

```
SELECT COUNT(nome) FROM produto;
+--------------+
| COUNT(nome) |
+--------------+
|            8 |
+--------------+
1 row in set (0.00 sec)

SELECT COUNT(id) FROM produto;
+------------+
| COUNT(id) |
+------------+
|         12 |
+------------+
1 row in set (0.00 sec)
```

#6

#6

Como trazer produtos com **nome** não preenchido?

#6

Como trazer produtos com **nome** não preenchido?

```
SELECT id, nome, valor
FROM produto
WHERE nome = '';
+----+------+--------+
| id | nome | valor |
+----+------+--------+
| 22 |      | 37.42 |
| 24 |      | 35.42 |
+----+------+--------+
2 rows in set (0.00 sec)
```

#6

Temos 4 produtos com o campo nome não preenchido. Onde estão eles?

#6

Temos 4 produtos com o campo nome não preenchido. Onde estão eles?

```
SELECT id, nome, valor
FROM produto
WHERE nome IS NULL;
+----+------+-------+
| id | nome | valor |
+----+------+-------+
| 21 | NULL | 38.42 |
| 28 | NULL | 31.42 |
| 31 | NULL |  9.32 |
| 32 | NULL |  8.45 |
+----+------+-------+
4 rows in set (0.00 sec)
```

#6

Alguns produtos estão com o campo EM BRANCO, outros estão NULOS (NULL).

Algumas produtos não foram editados, e continuam com o valor padrão do banco: NULL

Outros foram editados mas não tiveram o valor preenchido, estão agora com valor EM BRANCO.

#6

Agora toda query terá de fazer uma verificação dupla.

```
SELECT *
FROM produto
WHERE (nome = '' OR nome IS NULL);
+----+------+-------+----------+-------+
| id | nome | valor | estoque | ativo |
+----+------+-------+----------+-------+
| 21 | NULL | 38.42 |     4.00 |     1 |
| 22 |      | 37.42 |     8.00 |     1 |
| 24 |      | 35.42 |    11.00 |     1 |
| 28 | NULL | 31.42 |     4.00 |     1 |
| 31 | NULL |  9.32 |     2.00 |     1 |
| 32 | NULL |  8.45 |     3.00 |     1 |
+----+------+-------+----------+-------+
6 rows in set (0.00 sec)
```

#6

Agora toda query terá de fazer uma verificação dupla.

```
SELECT *
FROM produto
WHERE (nome = '' OR nome IS NULL);
+----+------+-------+----------+-------+
| id | nome | valor | estoque | ativo |
+----+------+-------+----------+-------+
| 21 | NULL | 38.42 |     4.00 |     1 |
| 22 |      | 37.42 |     8.00 |     1 |
| 24 |      | 35.42 |    11.00 |     1 |
| 28 | NULL | 31.42 |     4.00 |     1 |
| 31 | NULL |  9.32 |     2.00 |     1 |
| 32 | NULL |  8.45 |     3.00 |     1 |
+----+------+-------+----------+-------+
6 rows in set (0.00 sec)
```

#6

Dica: Usar o valor padrão. Usar o default:

```sql
UPDATE produto SET nome = '' WHERE nome IS NULL;



ALTER TABLE produto
  MODIFY COLUMN nome VARCHAR(100) DEFAULT '' NOT NULL;
```

#6

Dica: Usar o valor padrão. Usar o default:

```
UPDATE produto SET nome = '' WHERE nome IS NULL;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

ALTER TABLE produto
  MODIFY COLUMN nome VARCHAR(100) DEFAULT '' NOT NULL;
```

#6

Dica: Usar o valor padrão. Usar o default:

```
UPDATE produto SET nome = '' WHERE nome IS NULL;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

ALTER TABLE produto
  MODIFY COLUMN nome VARCHAR(100) DEFAULT '' NOT NULL;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

#7

#7 Dando valor pro nulo

Sabemos que NULO é NULO, VAZIO é VAZIO.

Nosso sistema permite o campo NULL, então temos alguns nulos no banco.

```
ALTER TABLE produto
  MODIFY COLUMN nome VARCHAR(100) DEFAULT '' NULL;

UPDATE produto
SET    nome = NULL
WHERE  nome = '';
```

#7

Trazer o nome do produto. Se ele for NULL, trazer o valor "Não informado".
Como trazer um valor padrão na hora de executar a query?

#7

Trazer o nome do produto. Se ele for NULL, trazer o valor "Não informado".
Como trazer um valor padrão na hora de executar a query?

```
SELECT id, nome, COALESCE(nome, "Não informado")
FROM   produto;
```

#7

Trazer o nome do produto. Se ele for NULL, trazer o valor "Não informado".

O COALESCE traz o primeiro valor não nulo do que passamos pra ele.

Isto é, se o nome for nulo, ele devolve “Não informado”.

Também podemos usar com números: COALESCE(valor, 0).

```
+----+----------+-------------------------------------+
| id | nome     | COALESCE(nome, "Não informado") |
+----+----------+-------------------------------------+
| 10 | Livro A  | Livro A                             |
| 11 | Livro B  | Livro B                             |
| 12 | Livro C  | Livro C                             |
| 13 | Livro D  | Livro D                             |
| 14 | Livro E  | Livro E                             |
| 15 | Livro F  | Livro F                             |
| 16 | Livro G  | Livro G                             |
| 17 | Livro H  | Livro H                             |
| 19 | Livro I  | Livro I                             |
| 20 | Livro J  | Livro J                             |
| 21 | NULL     | Não informado                       |
| 22 | NULL     | Não informado                       |
| 23 | Livro XA | Livro XA                            |
| 24 | NULL     | Não informado                       |
| 25 | Livro XB | Livro XB                            |
| 26 | Livro XC | Livro XC                            |
| 27 | Livro XD | Livro XD                            |
| 28 | NULL     | Não informado                       |
| 29 | Livro XE | Livro XE                            |
| 30 | Livro XF | Livro XF                            |
| 31 | NULL     | Não informado                       |
| 32 | NULL     | Não informado                       |
+----+----------+-------------------------------------+
22 rows in set (0.00 sec)
```

Tabela exemplo para usar com as próximas dicas

```sql
DROP TABLE IF EXISTS venda;
CREATE TABLE IF NOT EXISTS venda
  (
     id         INT auto_increment,
     produtora VARCHAR (20) DEFAULT '',
     valor      DECIMAL (10, 2) DEFAULT 0,
     PRIMARY KEY (id)
  );
```

```sql
INSERT INTO venda (produtora, valor) VALUES
  ('Microsoft', 100.00),
  ('Ubisoft', 99.00),
  ('Microsoft', 89.00),
  ('EA Games', 88.00),
  ('EA Games', 50.00),
  ('Microsoft', 80.00),
  ('', 0.00),
  ('', 100.00),
  ('Ubisoft', 0.00),
  ('EA Games', 100.00),
  ('Microsoft', 100.00),
  ('Microsoft', 100.00),
  ('EA Games', NULL),
  ('Microsoft', 0.00),
  ('EA Games', 100.00),
  ('', 0.00),
  ('Microsoft', 100.00),
  ('Ubisoft', 100.00),
  ('Ubisoft', 100.00),
  (NULL, NULL),
  ('Microsoft', 100.00),
  (NULL, 100.00),
  ('Microsoft', 100.00),
  ('Ubisoft', 100.00),
  ('Ubisoft', 100.00),
  (NULL, 100.00);
```


#8

#8

Numa consulta para saber como estão as vendas por produtora, agrupamos:

```
SELECT produtora, SUM(valor) AS total
FROM venda
GROUP BY produtora
ORDER BY produtora;
+-------------+--------+
| produtora | total  |
+-------------+--------+
| NULL        | 200.00 |
|             | 100.00 |
| EA Games    | 338.00 |
| Microsoft | 769.00 |
| Ubisoft     | 499.00 |
+-------------+--------+
5 rows in set (0.00 sec)
```

#8

Numa consulta para saber como estão as vendas por produtora, agrupamos:

Tem vários resultados "menores" que não estou tão interessado.
Só quero enxergar quem vendeu mais de **400**.

```
SELECT produtora, SUM(valor) AS total
FROM venda
GROUP BY produtora
ORDER BY produtora;
+-------------+--------+
| produtora | total  |
+-------------+--------+
| NULL        | 200.00 |
|             | 100.00 |
| EA Games    | 338.00 |
| Microsoft | 769.00 |
| Ubisoft     | 499.00 |
+-------------+--------+
5 rows in set (0.00 sec)
```

#8

Usando subconsuta:

#8

Usando subconsuta:

```
SELECT produtora,
       total
FROM   (SELECT produtora,
               SUM(valor) AS total
        FROM   venda
        GROUP  BY produtora) AS temp
WHERE  total > 400
ORDER BY produtora;
+------------+--------+
| produtora | total  |
+------------+--------+
| Microsoft | 769.00 |
| Ubisoft   | 499.00 |
+------------+--------+
2 rows in set (0.00 sec)
```

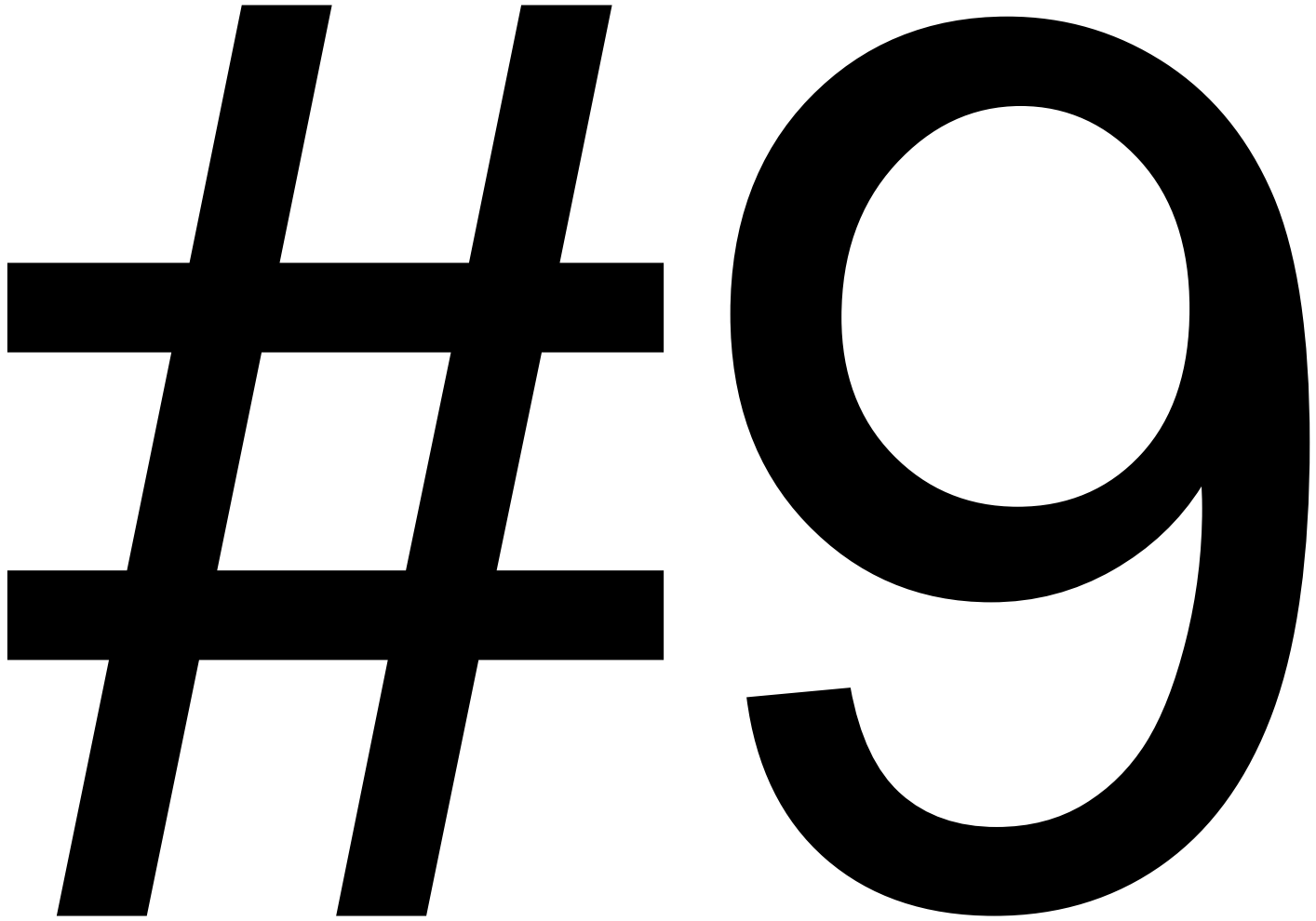
#8

Mesmo resultado sem usar subconsulta
(usando o HAVING)

#8

Mesmo resultado sem usar subconsulta (usando o HAVING)

```
SELECT produtora,
       SUM(valor) AS vendas
FROM   venda
GROUP  BY produtora
HAVING vendas > 400
ORDER BY produtora;
+-------------+--------+
| produtora | vendas |
+-------------+--------+
| Microsoft | 769.00 |
| Ubisoft   | 499.00 |
+-------------+--------+
2 rows in set (0.00 sec)
```


#9

#9

Quais são as minhas tabelas?

```sql
-- MySQL
SHOW TABLES; -- (Que barbada esse tal de MySQL)

-- Microsoft SQL Server
SELECT *
FROM   information_schema.TABLES;

-- ORACLE - Minhas tabelas
SELECT table_name
FROM   user_tables;

-- ORACLE - Todas tabelas que tenho acesso
SELECT table_name
FROM   all_tables;

-- POSTGRESQL
SELECT *
FROM   pg_catalog.pg_tables;
```

#10

Usando TRIGGER, no MySQL, para impedir inserção de registro (de acordo com uma determinada condição)

Passo 1: Criar uma tabela com uma coluna que não aceite valores NULL

```sql
DROP TABLE IF EXISTS alunos;
CREATE TABLE IF NOT EXISTS alunos(
  id INT NOT NULL,
  nome VARCHAR(45)
);
```

Passo 2: Criar TRIGGER para conferir o valor antes do INSERT

```
DELIMITER $$
CREATE TRIGGER TRG_bloqueador
BEFORE INSERT
ON alunos
FOR EACH ROW
BEGIN
  IF (NEW.id < 10) THEN
    SET NEW.id = NULL;
  END IF;
END;
$$
DELIMITER ;
```

Passo 3: Teste de inserção

```
INSERT INTO alunos (id, nome) VALUES (1, 'Tiririca');
ERROR 1048 (23000): Column 'id' cannot be null

INSERT INTO alunos (id, nome) VALUES (33, 'Tiririca');
Query OK, 1 row affected (0.00 sec)
```

Passo 4: Criando STORED PROCEDURE para fazer o INSERT (tratando ERRO 23000)

```
DELIMITER $$
CREATE PROCEDURE SP_legal (v_id INT, v_nome VARCHAR(45))
BEGIN
  DECLARE EXIT HANDLER FOR SQLSTATE '23000'
  BEGIN
    SELECT 'INCLUSÃO NÃO EFETUADA' AS MSG;
  END;
  INSERT INTO alunos (id, nome) VALUES (v_id, v_nome);
  SELECT 'INCLUSÃO OK' AS MSG;
END;
$$
DELIMITER ;
```

Passo 5: Teste de inserção através de STORED Procedure

```
CALL SP_legal (1, 'Tiririca');
+-------------------------+
| MSG                     |
+-------------------------+
| INCLUSÃO NÃO EFETUADA |
+-------------------------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)

CALL SP_legal (33, 'Tiririca');
+---------------+
| MSG           |
+---------------+
| INCLUSÃO OK |
+---------------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

+Dicas

ALIASES

Função para “apelidar” tabelas dentro do SQL:

Ex.: 1

```
SELECT a.nome, d.nome
FROM aluno a
INNER JOIN, disciplina d
WHERE a.mat > 30 AND d.nome=“BDD”;
```

2

Quais os livros escritos pelo autor de código 7009X?

```
SELECT l.id
FROM livro l, livroAutor la
WHERE l.id = la.livro_id
AND la.autor_id = “7009X”;
```

SELF JOINS

Você pode utilizar o JOIN usando a mesma tabela, dando dois aliases, fazendo comparações dentro da mesma:

Ex.:

```
SELECT a1.Snome, a1.Pnome
FROM autor a1
INNER JOIN autor a2 ON a1.Snome = a2.Snome
WHERE a1.codigo!= a2.codigo;
```

Que autores possuem o mesmo sobrenome?

CLÁUSULA DISTINCT

Com exceção da chave primária, podem existir colunas que tenham valores repetitivos. Essa cláusula aplicada em uma consulta evita valores repetitivos dentro de uma consulta.

Ex.:

```
SELECT DISTINCT nome
FROM produto;
```

LIKE e NOT LIKE

Só funcionam com colunas do tipo char;

Têm praticamente o mesmo funcionamento que os operadores **=** e **!=**;

Sua vantagem é a utilizações dos símbolos:

%: substitui uma palavra;

_: substitui um caracter.

LIKE e NOT LIKE

Exemplos:

Listar todos os produtos cujo nome comece com Q.

```
SELECT codigo_produto, descricao_produto
FROM produto
WHERE descricao_produto LIKE ’Q%’;
```

LIKE e NOT LIKE

Mostrar os professores que o primeiro nome seja João.

```
SELECT codigo, nome
FROM professor
WHERE nome LIKE ‘João%’;
```

Operadores baseados em IS NULL e IS NOT NULL

Mostrar os empregados que tenham seus salários cadastrados no sistema como NULO.

```
SELECT nome FROM empregado
WHERE salario IS NULL;
```

Mostrar as disciplinas que tenham a carga horária como não nulo.

```
SELECT nome, codigo FROM disciplina
WHERE carga_horaria IS NOT NULL;
```

MAX e MIN

MAX: mostra o maior valor dentro de um campo em uma tabela;

MIN: mostra o menor valor de um campo dentro de uma tabela;

Ex.:

```
SELECT MIN(salario_fixo), MAX(salario_fixo)
FROM vendedor;
```

SUM

Serve para fazer o somatório de todos os valores de uma coluna.

Ex.:

```
SELECT SUM(QUANTIDADE)
FROM item_pedido
WHERE codigo_produto = ’50’;
```

AVG

Apresenta a média de uma coluna.

Ex.: Qual a média dos salários fixos dos vendedores?

```
SELECT AVG(salario_fixo)
FROM vendedor;
```

MySQL
VIEWS

VIEW

É um objeto baseado em declarações SELECT’s, retornando uma determinada visualização de dados de uma ou mais tabelas.

E alguns casos, as Views são atualizáveis e podem ser alvos de declaração INSERT, UPDATE e DELETE, que na verdade modificam sua “based tables”.

Visualizar as Views de um banco:

```
SELECT table_name
FROM information_schema.views
```

Como criar uma View

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

*verde – cláusula opcional

Como criar uma View

Exemplo:

```
CREATE OR replace VIEW vw_alunos
AS
   SELECT matricula,
          nome
   FROM   alunos;
```

Mais sobre Views

Podemos criar Views sofisticadas, com comandos SELECT, podendo utilizar cláusulas WHERE, GROUP BY, HAVING e ORDER BY.

Alguns SGBD’s comerciais não permitem a utilização de ORDER BY em meio ao SELECT na definição de uma View, a exemplo do SQL Server, da Microsoft.

Pivotamento