**Banco de Dados – Comandos DML: Manipulação de Dados com MySQL– Micro-certificação - ALURA**

# Aula 01 – Modelagem do Banco de Dados

1) Instale o MySQL, conforme o vídeo [**Instalando o MySQL Server**](https://cursos.alura.com.br/course/mysql-manipule-dados-com-sql/task/54324), da aula 1 do curso [**Introdução ao SQL: Manipule dados com MySQL**](https://cursos.alura.com.br/course/mysql-manipule-dados-com-sql).

2) Abra o MYSQL Workbench. Use a conexão LOCALHOST e certifique que o ambiente está funcionando.

# O que aprendemos?

* Como modelar um banco de dados baseado no negócio da empresa;
* Revisamos todas as entidades envolvidas em um banco de dados;
* Aprendemos os tipos de dados que compõem uma tabela.

### Aula 02 – Criando a estrutura do banco de dados

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Abra o MYSQL Workbench e crie um novo script de comandos SQL.

2) Para criar uma base de dados podemos usar o comando CREATE. Digite o comando abaixo para criar uma nova base de dados:

CREATE DATABASE vendas\_sucos;

3) Podemos usar algumas propriedades para definir o tipo de tabela de caracteres que podem ser usadas na armazenagem dos dados no banco. Digite o comando abaixo para criar um banco que utilize a tabela de caracteres UTF8:

CREATE DATABASE vendas\_sucos2 DEFAULT CHARACTER SET utf8;

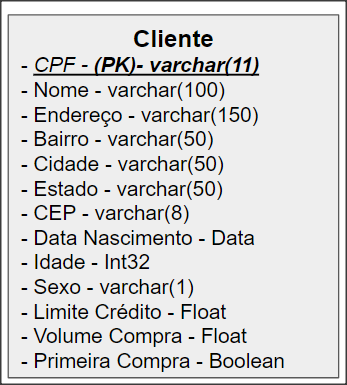
4) Se tivermos dúvida se o banco de dados a ser criado existe ou não, podemos usar o comando IF NOT EXISTS. Este comando é usado em diversas entidades do MYSQL e é usado para que o processo de criação aconteça apenas se o componente existe. Digite o comando abaixo para criar novo banco, somente se ele não existir:

CREATE DATABASE IF NOT exists vendas\_sucos2;

5) Podemos apagar um banco de dados existente. Para isso digite o comando:

DROP DATABASE IF EXISTS vendas\_sucos2;

6) Clique com o botão da direita do mouse sobre a área vazia na lista de bases de dados, a direita no Workbech, e escolha a opção Create Schema.

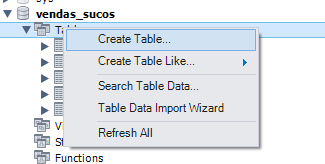


7) Será apresentado um assistente que lhe ajuda a criar um banco de dados sem precisar usar os comandos. Digite no nome da base vendas\_sucos2 e escolha as opções UTF16 para lista de caracteres.



8) Clique em Apply. O comando SQL a ser executado é gerado dinamicamente. Confirme a execução e o banco será criado.

9) Clique com o botão da direita do mouse sobre vendas\_sucos2 e escolha a opção DROP SCHEMA.



A base será excluida com este comando.

10) Duplo clique sobre a base de dados vendas\_sucos para que a mesma fique selecionada. O nome dela deverá ficar em negrito.

11) Crie um novo script de SQL e digite o comando:

USE vendas\_sucos;

Este comando forçará a seleção da base de dados vendas\_sucos;

12) Vamos criar a primeira tabela no nosso banco. Digite e execute:

CREATE TABLE PRODUTOS

(CODIGO VARCHAR(10) NOT NULL,

DESCRITOR VARCHAR(100) NULL,

SABOR VARCHAR(50) NULL,

TAMANHO VARCHAR(50) NULL,

EMBALAGEM VARCHAR(50) NULL,

PRECO\_LISTA FLOAT NULL,

PRIMARY KEY (CODIGO));

13) Vamos criar a segunda tabela (vendedores). Digite:

CREATE TABLE VENDEDORES

(MATRICULA VARCHAR(5) NOT NULL,

NOME VARCHAR(100) NULL,

BAIRRO VARCHAR(50) NULL,

COMISSAO FLOAT NULL,

DATA\_ADIMISSAO DATE NULL,

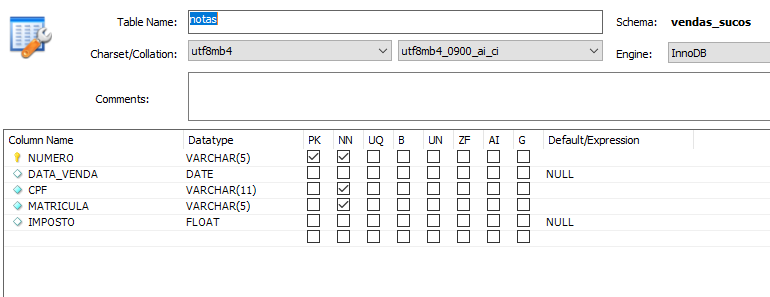
FERIAS BIT(1) NULL,

PRIMARY KEY (MATRICULA));

14) Note que houve um erro proposital ao selecionar o nome do campo que mostra a data de admissão do vendedor. Podemos alterar este nome mesmo com a tabela criada. Para isso digite execute::

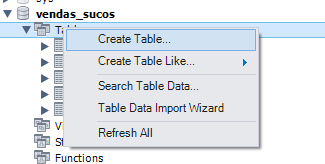
ALTER TABLE VENDEDORES RENAME COLUMN DATA\_ADIMISSAO TO DATA\_ADMISSAO;

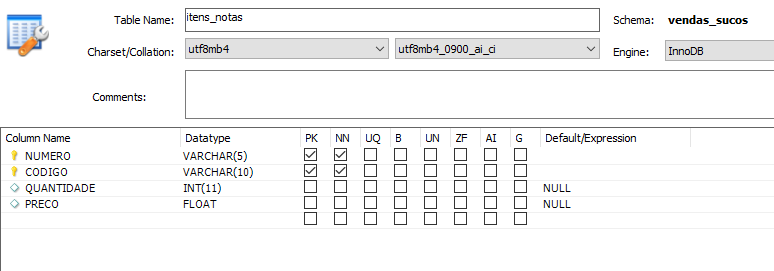
15) Clique com o botão da direita do mouse sobre o nó Table e selecione a opção Create Table.



16) Temos a caixa de diálogo para criar uma tabela através de um assistente. Digite na caixa de diálogo:







17) Confirme a criação da tabela acima.

18) Vamos criar mais uma tabela que é a tabela de vendas (Cabeçário da nota fiscal). Para isso digite execute:

USE vendas\_sucos;

CREATE TABLE TABELA\_DE\_VENDAS

(NUMERO VARCHAR(5) NOT NULL,

DATA\_VENDA DATE NULL,

CPF VARCHAR(11) NOT NULL,

MATRICULA VARCHAR(5) NOT NULL,

IMPOSTO FLOAT NULL,

PRIMARY KEY (NUMERO));

19) Podemos criar relacionamentos entre esta tabela e a tabela de clientes e vendedores. Para isso digite e execute:

ALTER TABLE TABELA\_DE\_VENDAS ADD CONSTRAINT FK\_CLIENTES

FOREIGN KEY (CPF) REFERENCES CLIENTES (CPF);

ALTER TABLE TABELA\_DE\_VENDAS ADD CONSTRAINT FK\_VENDEDORES

FOREIGN KEY (MATRICULA) REFERENCES VENDEDORES (MATRICULA);

20) Da mesma maneira que fizemos em alguns passos anteriores, com o campo, o nome da tabela também pode ser alterado depois da mesma ser criada. Digite execute:

USE vendas\_sucos;

ALTER TABLE tabela\_de\_vendas RENAME Notas;

# O que aprendemos?

Nesta aula, aprendemos:

* Aprendemos a criar o banco de dados.
* Aprendemos a criar uma tabela;
* Vimos que o nome dos campos podem ser modificados mesmo depois da tabela ter sido criada;
* Podemos criar as tabelas por um assistente. Foi mostrado isso nesta aula;
* Vimos como criar os relacionamentos entre as tabelas;
* Vimos que o nome da tabela também pode ser modificada após sua criação.

### Aula 03 – Incluindo Dados nas Tabelas

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Vamos criar um novo script SQL no Workbench.

2) Iremos inserir um novo produto. Digite:

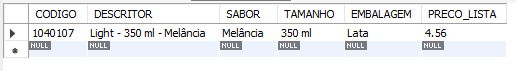
USE vendas\_sucos;

INSERT INTO PRODUTOS (CODIGO, DESCRITOR, SABOR, TAMANHO, EMBALAGEM, PRECO\_LISTA)

VALUES ('1040107', 'Light - 350 ml - Melância', 'Melância', '350 ml', 'Lata', 4.56);

3) Vamos conferir se o produto foi realmente incluido na tabela. Digite e execute:

SELECT \* FROM PRODUTOS;



4) No script podemos incluir mais de um comando. Digite e execute:

INSERT INTO PRODUTOS (CODIGO, DESCRITOR, SABOR, TAMANHO, EMBALAGEM, PRECO\_LISTA)

VALUES ('1040108', 'Light - 350 ml - Graviola', 'Graviola', '350 ml', 'Lata', 4.00);

INSERT INTO PRODUTOS

VALUES ('1040109', 'Light - 350 ml - Açai', 'Açai', '350 ml', 'Lata', 5.60);

5) Também podemos incluir, num mesmo comando, a inclusão de mais de um registro. Digite e execute:

INSERT INTO PRODUTOS

VALUES ('1040110', 'Light - 350 ml - Jaca', 'Jaca', '350 ml', 'Lata', 6.00),

('1040111', 'Light - 350 ml - Manga', 'Manga', '350 ml', 'Lata', 3.50);

6) Faça Download do arquivo RecuperacaoAmbiente.zip.

7) Descompacte o arquivo.

8) Selecione, na área Navigator, em Administration.

9) Selecione o link Data Import/Restore.

10) Na opção Import From Dump Project Folder escolha o diretório DumpSucosVendas.

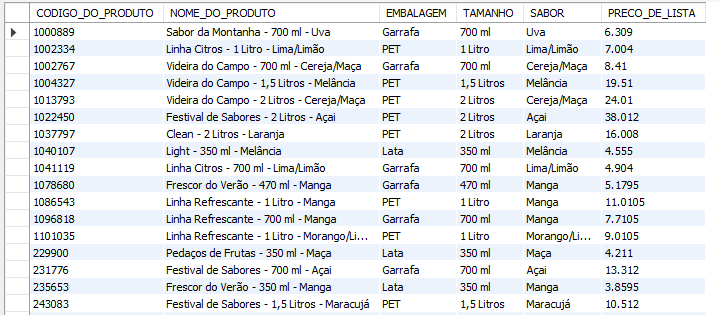
11) Selecione Start Import.

12) Verifique na base Sucos\_Vendas se as tabelas foram criadas.

13) Podemos, da base Vendas\_Sucos, acessar tabelas da base Sucos\_Vendas.

USE vendas\_sucos;

SELECT \* FROM sucos\_vendas.tabela\_de\_produtos;



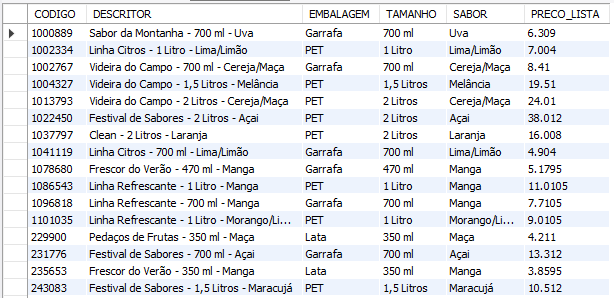
14) A consulta, a seguir, mostra a lista de produtos, na tabela tabela\_de\_produtos, da base sucos\_vendas que ainda não foram incluídas na tabela produtos, da base vendas\_sucos:

SELECT CODIGO\_DO\_PRODUTO AS CODIGO, NOME\_DO\_PRODUTO AS DESCRITOR,

EMBALAGEM, TAMANHO, SABOR, PRECO\_DE\_LISTA AS PRECO\_LISTA

FROM sucos\_vendas.tabela\_de\_produtos

WHERE CODIGO\_DO\_PRODUTO NOT IN (SELECT CODIGO FROM produtos);



15) No comando INSERT podemos incluir os produtos na tabela sucos\_vendas.tabela\_de\_produtos na tabela vendas\_sucos.produtos. Digite e execute:

INSERT INTO produtos

SELECT CODIGO\_DO\_PRODUTO AS CODIGO, NOME\_DO\_PRODUTO AS DESCRITOR,

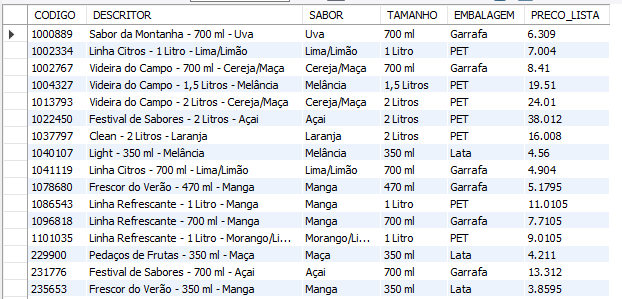
SABOR, TAMANHO, EMBALAGEM, PRECO\_DE\_LISTA AS PRECO\_LISTA

FROM sucos\_vendas.tabela\_de\_produtos

WHERE CODIGO\_DO\_PRODUTO NOT IN (SELECT CODIGO FROM produtos);

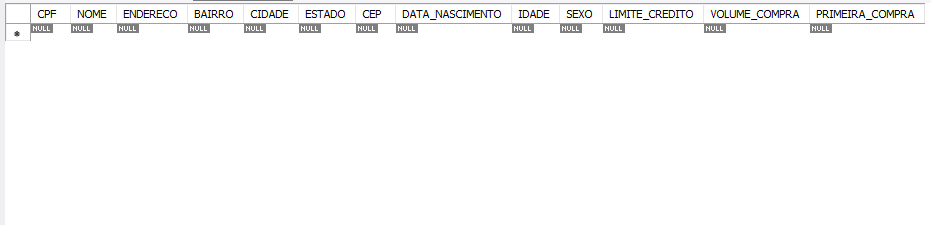
16) Vamos conferir a tabela de produtos. Digite e execute:

SELECT \* FROM produtos;



17) Mostraremos como incluir dados na tabela de cliente. Digite e execute:

SELECT \* FROM clientes;



18) Ao lado temos o botão Form Editor:



Teremos uma caixa de diálogo para editar a tabela de Clientes.



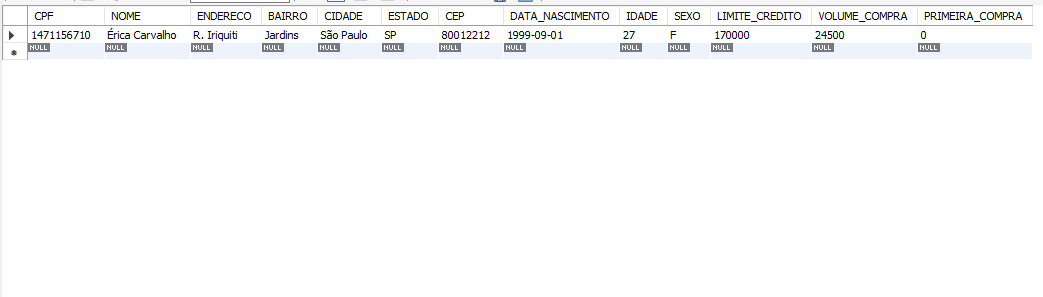
19) Inclua um novo cliente:

* CPF: 1471156710;
* NOME: Érica Carvalho;
* ENDERECO: R. Iriquiti;
* BAIRRO: Jardins;
* CIDADE: São Paulo;
* ESTADO: SP;
* CEP: 80012212;
* DATA\_NASCIMENTO: 1999-09-01;
* IDADE: 27;
* SEXO: F;
* LIMITE\_CREDITO: 170000;
* VOLUME\_COMPRA: 24500;
* PRIMEIRA\_COMPRA: 0;

20) Confirme a inclusão. O comando será apresentado e confirme a execução.

21) Execute a consulta novamente. Digite e execute:

SELECT \* FROM clientes;



O cliente foi incluido.

22) Vamos incluir os clientes usando como fonte o outro banco de dados. Digite e execute:

INSERT INTO CLIENTES

SELECT CPF, NOME, ENDERECO\_1 AS ENDERECO, BAIRRO, CIDADE, ESTADO, CEP,

DATA\_DE\_NASCIMENTO AS DATA\_NASCIMENTO, IDADE, SEXO, LIMITE\_DE\_CREDITO AS LIMITE\_CREDITO,

VOLUME\_DE\_COMPRA AS VOLUME\_COMPRA, PRIMEIRA\_COMPRA

FROM sucos\_vendas.tabela\_de\_clientes WHERE CPF

NOT IN (SELECT CPF FROM CLIENTES);

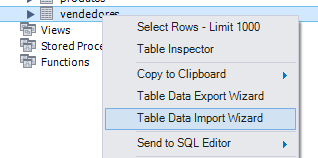
23) Teste o conteúdo da tabela de clientes:

SELECT \* FROM CLIENTES;



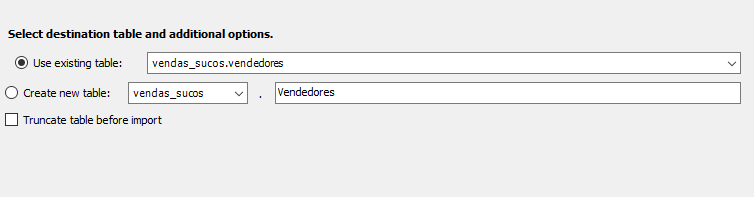
24) Vamos mostrar agora como incluir dados na tabela de vendedores através de arquivos externos. Descompacte o arquivo contido no link desta aula e iremos usar a tabela Vendedores.csv. como fonte.

25) Botão da direita do mouse sobre a tabela Vendedores e escolha a opção Table Data Import Wizard.

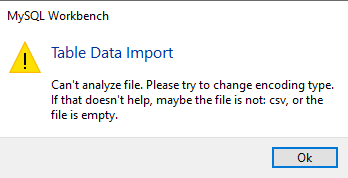


26) Em File Path selecione o arquivo Vendedores.csv.

27) Mantenha os dados padrões como mostrado abaixo:



28) Observação: Se você visualizar um erro como mostrado abaixo:

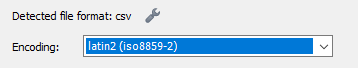


Há erro na interpretação da tabela através da lista de caracteres. Para isso faça:

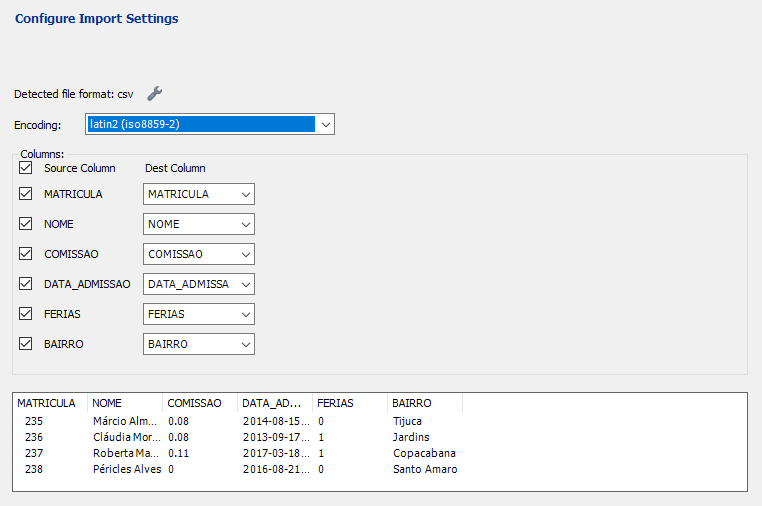
* Abra o arquivo com Notepad clássico;
* Clique arquivo / Salvar como.
* Escolha o padrão ANSI:

13.png

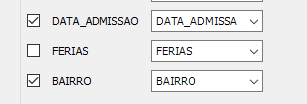
Volte a caixa de diálogo e escolha:



Pode existir problemas entre computadores e entre os arquivos baixados. Você deve ver a combinação correta entre o formato do arquivo (Que pode ser modificado no NOTEPAD clássico) e na caixa de diálogo de importação de dados.



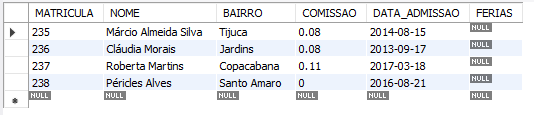
29) Desmarque a opção FERIAS:



30) Clique em Next várias vezes até os dados serem incluídos na tabela de vendedores.

31) Verifique o conteúdo da tabela de vendedores.

SELECT \* FROM VENDEDORES;



# O que aprendemos?

Nesta aula, aprendemos:

* Aprendemos a incluir dados em uma tabela, através de comando;
* Vimos a inclusão de vários registros em um único comando;
* Foi mostrado como incluir dados na tabela através de um assistente;
* Mostramos como incluir dados usando um arquivo externo.

### Aula 04 – Alterando e Excluindo Dados Existentes nas Tabelas

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Vamos verificar a lista de produtos.

SELECT \* FROM produtos;



2) Vamos alterar o preço de lista de um dos produtos. Para isso digite:

UPDATE produtos SET PRECO\_LISTA = 5 WHERE CODIGO = '1000889';

3) Podemos alterar os dados da tabela em forma de lote. Digite e execute:

UPDATE produtos SET EMBALAGEM = 'PET', TAMANHO = '1 Litro', DESCRITOR =

'Sabor da Montanha - 1 Litro - Uva' WHERE CODIGO = '1000889';

4) Também podemos alterar o preço de lista baseado no mesmo campo que será alterado. Digite e execute:

UPDATE produtos SET PRECO\_LISTA = PRECO\_LISTA \* 1.10 WHERE SABOR = 'Maracujá';

5) Da mesma maneira que incluímos dados na tabela baseado nos dados de uma outra tabela, podemos também alterar dados desta mesma maneira. Digite e execute:

UPDATE VENDEDORES A INNER JOIN SUCOS\_VENDAS.TABELA\_DE\_VENDEDORES B

ON A.MATRICULA = SUBSTRING(B.MATRICULA,3,3)

SET A.FERIAS = B.DE\_FERIAS;

6) É possível apagar dados da tabela. Antes disto vamos incluir novos registros que depois serão excluídos. Digite e execute:

INSERT INTO PRODUTOS (CODIGO,DESCRITOR,SABOR,TAMANHO,EMBALAGEM,PRECO\_LISTA)

VALUES ('1001001','Sabor dos Alpes 700 ml - Manga','Manga','700 ml','Garrafa',7.50),

('1001000','Sabor dos Alpes 700 ml - Melão','Melão','700 ml','Garrafa',7.50),

('1001002','Sabor dos Alpes 700 ml - Graviola','Graviola','700 ml','Garrafa',7.50),

('1001003','Sabor dos Alpes 700 ml - Tangerina','Tangerina','700 ml','Garrafa',7.50),

('1001004','Sabor dos Alpes 700 ml - Abacate','Abacate','700 ml','Garrafa',7.50),

('1001005','Sabor dos Alpes 700 ml - Açai','Açai','700 ml','Garrafa',7.50),

('1001006','Sabor dos Alpes 1 Litro - Manga','Manga','1 Litro','Garrafa',7.50),

('1001007','Sabor dos Alpes 1 Litro - Melão','Melão','1 Litro','Garrafa',7.50),

('1001008','Sabor dos Alpes 1 Litro - Graviola','Graviola','1 Litro','Garrafa',7.50),

('1001009','Sabor dos Alpes 1 Litro - Tangerina','Tangerina','1 Litro','Garrafa',7.50),

('1001010','Sabor dos Alpes 1 Litro - Abacate','Abacate','1 Litro','Garrafa',7.50),

('1001011','Sabor dos Alpes 1 Litro - Açai','Açai','1 Litro','Garrafa',7.50);

7) Vamos apagar um registro apenas. Digite e execute:

DELETE FROM PRODUTOS WHERE CODIGO = '1001000';

8) Podemos aplicar um filtro para selecionar dados a serem excluidos. Digite e execute:

DELETE FROM PRODUTOS WHERE TAMANHO = '1 Litro' AND SUBSTRING(DESCRITOR,1,15) = 'Sabor dos Alpes';

9) Outra forma é apagar usando a seleção de dados de outra tabela. Digite e execute:

DELETE FROM PRODUTOS WHERE CODIGO NOT IN ( SELECT CODIGO\_DO\_PRODUTO FROM SUCOS\_VENDAS.TABELA\_DE\_PRODUTOS);

10) O comando UPDATE e DELETE poser ser executado sobre a tabela inteira. Vamso então criar uma tabela de forma temporária para depois alterá-la e apagá-la. Digite e execute:

CREATE TABLE `produtos2` (

`CODIGO` varchar(10) NOT NULL,

`DESCRITOR` varchar(100) DEFAULT NULL,

`SABOR` varchar(50) DEFAULT NULL,

`TAMANHO` varchar(50) DEFAULT NULL,

`EMBALAGEM` varchar(50) DEFAULT NULL,

`PRECO\_LISTA` float DEFAULT NULL,

PRIMARY KEY (`CODIGO`)

) ;

11) Depois inclua os dados nesta tabela:

INSERT INTO produtos2 SELECT \* FROM produtos;

12) Altere os dados para toda a tabela:

UPDATE produtos2 SET preco\_lista = 8;

13) O comando abaixo apaga todos os registros da tabela. Digite o comando abaixo e execute para apagar os dados da tabela:

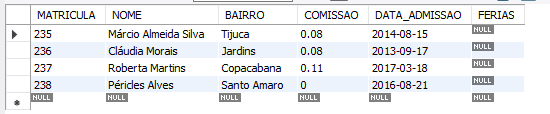
DELETE FROM produtos2;

14) Vamos criar uma transação. Digite e execute o comando abaixo para iniciar uma transação:

START TRANSACTION;

15) Vamos verificar a tabela de vendedores:

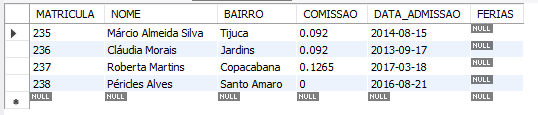
SELECT \* FROM VENDEDORES;



16) Modifique os dados referentes a comissão. Digite e execute:

UPDATE VENDEDORES SET COMISSAO = COMISSAO \* 1.15;

SELECT \* FROM VENDEDORES;



17) Vamos refazer o comando acima. Digite e execute:

ROLLBACK;

18) Verifique a tabela. Digite e execute:

SELECT \* FROM VENDEDORES;

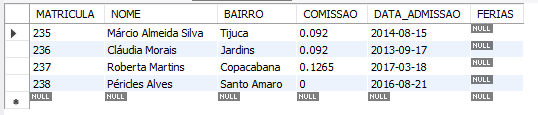


19) Vamos repetir a modificação iniciando nova transação. Digite e execute:

START TRANSACTION;

UPDATE VENDEDORES SET COMISSAO = COMISSAO \* 1.15;

SELECT \* FROM VENDEDORES;

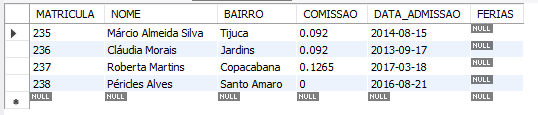


20) Agora iremos confirmar a inclusão destes dados. Para isso digite:

COMMIT;

21) Verifique novamente a tabela. Para isso digite:

SELECT \* FROM VENDEDORES;



# O que aprendemos?

Nesta aula, aprendemos:

* Alterar e excluir dados de uma tabela, através de comando ou em lotes;
* Vimos que podemos alterar ou excluir todos os dados de uma tabela;
* O que é uma transação e como podemos desfazer modificações efetuadas anteriormente na base de dados.

### Aula 05 – Auto Incremento, Padrões e Triggers

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com os próximos cursos que tenham este como pré-requisito.

1) O campo do tipo auto incremento cria uma sequência numérica de números inteiros em um campo. Para definir este campo precisamos configurá-lo na criação da tabela. Logo digite o comando abaixo e execute:

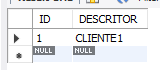
CREATE TABLE TAB\_IDENTITY (ID INT AUTO\_INCREMENT, DESCRITOR VARCHAR(20), PRIMARY KEY(ID));

2) Para inserir um registro não precisamos nos referenciar ao campo auto incremento no comando **INSERT**. Digite e execute:

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE1');

3) Verifique o conteúdo da tabela. Digite e execute:

SELECT \* FROM TAB\_IDENTITY;



4) Abaixo vemos diversas formas de inclusão de novos registros. Digite e execute:

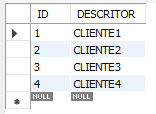
INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE2');

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE3');

INSERT INTO TAB\_IDENTITY (ID, DESCRITOR) VALUES (NULL, 'CLIENTE4');

5) Verifique o conteúdo da tabela. Digite e execute:

SELECT \* FROM TAB\_IDENTITY;

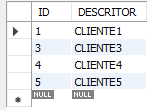


6) Ao apagar um registro não interrompemos a sequência do contador. veja também que, se quisermos manter o campo auto incremento no comando **INSERT** temos que referencia-lo com null para não interromper a sequência. Digite e execute:

DELETE FROM TAB\_IDENTITY WHERE ID = 2;

INSERT INTO TAB\_IDENTITY (ID, DESCRITOR) VALUES (NULL, 'CLIENTE5');

SELECT \* FROM TAB\_IDENTITY;



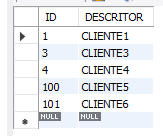
7) Caso a gente force um valor para o campo auto incremento a sequência será re- atualizada. Digite e execute:

INSERT INTO TAB\_IDENTITY (ID, DESCRITOR) VALUES (100, 'CLIENTE5');

DELETE FROM TAB\_IDENTITY WHERE ID = 5;

INSERT INTO TAB\_IDENTITY (ID, DESCRITOR) VALUES (NULL, 'CLIENTE6');

SELECT \* FROM TAB\_IDENTITY;



8) Podemos definir padrões para os campos. Com isto um campo pode ter um valor default caso não seja referenciado no comando **INSERT**. Digite e execute:

CREATE TABLE TAB\_PADRAO

(ID INT AUTO\_INCREMENT,

DESCRITOR VARCHAR(20),

ENDERECO VARCHAR(100) NULL,

CIDADE VARCHAR(50) DEFAULT 'Rio de Janeiro',

DATA\_CRIACAO TIMESTAMP DEFAULT CURRENT\_TIMESTAMP(),

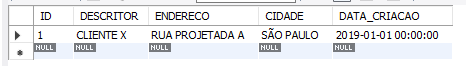
PRIMARY KEY(ID));

9) Padrões foram criados para o campo **ENDERECO**, **CIDADE** e **DATA\_CRIACAO**. Digite e execute:

INSERT INTO TAB\_PADRAO (DESCRITOR, ENDERECO, CIDADE, DATA\_CRIACAO)

VALUES ('CLIENTE X', 'RUA PROJETADA A', 'SÃO PAULO', '2019-01-01');

SELECT \* FROM TAB\_PADRAO;

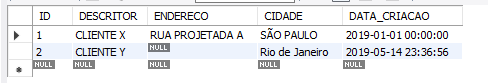


Aqui o comando **INSERT** funciona normalmente porque todos os campos foram referenciados.

10) Vamos repetir o comando **INSERT**, mas agora usando somente os campos que não possuem padrões. Digite e execute:

INSERT INTO TAB\_PADRAO (DESCRITOR) VALUES ('CLIENTE Y');

SELECT \* FROM TAB\_PADRAO;



Note que os campos que não foram referenciados no comando **INSERT** os seus valores padrões foram incluídos na tabela.

11) Vamos criar uma tabela auxiliar que irá sempre ter o faturamento consolidado por data da venda. Execute o comando:

CREATE TABLE TAB\_FATURAMENTO

(DATA\_VENDA DATE NULL, TOTAL\_VENDA FLOAT);

12) O objetivo é que, a cada inclusão de dados na tabela de **NOTAS** e **ITENS\_NOTAS** o valor da **TAB\_FATURAMENTO** seja atualizado. Para isso digite e execute:

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0100', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0100', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0100', '1002334', 100, 10);

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0101', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0101', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0101', '1002334', 100, 10);

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0103', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0103', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0103', '1002334', 100, 10);

13) Para atualizarmos a tabela **TAB\_FATURAMENTO** temos que executar o comando abaixo. Para isso digite e execute:

INSERT INTO TAB\_FATURAMENTO

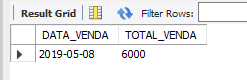
SELECT A.DATA\_VENDA, SUM(B.QUANTIDADE \* B.PRECO) AS TOTAL\_VENDA FROM

NOTAS A INNER JOIN ITENS\_NOTAS B

ON A.NUMERO = B.NUMERO

GROUP BY A.DATA\_VENDA;

SELECT \* FROM TAB\_FATURAMENTO;



14) Se queremos manter a tabela **TAB\_FATURAMENTO** atualizada temos que repetir sempre o cálculo atual do valor total das vendas sempre que novos registros forem incluídos. Para isso digite e execute:

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0104', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0104', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0104', '1002334', 100, 10);

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0105', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0105', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0105', '1002334', 100, 10);

DELETE FROM TAB\_FATURAMENTO;

INSERT INTO TAB\_FATURAMENTO

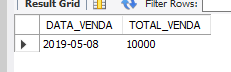
SELECT A.DATA\_VENDA, SUM(B.QUANTIDADE \* B.PRECO) AS TOTAL\_VENDA FROM

NOTAS A INNER JOIN ITENS\_NOTAS B

ON A.NUMERO = B.NUMERO

GROUP BY A.DATA\_VENDA;

SELECT \* FROM TAB\_FATURAMENTO;



15) Podemos criar uma **TRIGGER** para que a tabela **TAB\_FATURAMENTO** seja recalculada sempre que um novo registro for incluído na tabela de **ITENS\_NOTAS**. Para isso digite e execute:

DELIMITER //

CREATE TRIGGER TG\_CALCULA\_FATURAMENTO\_INSERT AFTER INSERT ON ITENS\_NOTAS

FOR EACH ROW BEGIN

DELETE FROM TAB\_FATURAMENTO;

INSERT INTO TAB\_FATURAMENTO

SELECT A.DATA\_VENDA, SUM(B.QUANTIDADE \* B.PRECO) AS TOTAL\_VENDA FROM

NOTAS A INNER JOIN ITENS\_NOTAS B

ON A.NUMERO = B.NUMERO

GROUP BY A.DATA\_VENDA;

END//

16) Ao inserir novos registros não é mais preciso executar o cálculo da tabela consolidada. Para isso digite e execute:

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0106', '2019-05-08', '1471156710' , '235', 0.10);

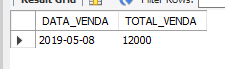
INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0106', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0106', '1002334', 100, 10);

SELECT \* FROM TAB\_FATURAMENTO;



17) Porém foi criado uma **TRIGGER** somente para inclusão de registros na tabela. Vamos incluir **TRIGGERS** para a atualização e exclusão. Para isso digite e execute:

DELIMITER //

CREATE TRIGGER TG\_CALCULA\_FATURAMENTO\_UPDATE AFTER UPDATE ON ITENS\_NOTAS

FOR EACH ROW BEGIN

DELETE FROM TAB\_FATURAMENTO;

INSERT INTO TAB\_FATURAMENTO

SELECT A.DATA\_VENDA, SUM(B.QUANTIDADE \* B.PRECO) AS TOTAL\_VENDA FROM

NOTAS A INNER JOIN ITENS\_NOTAS B

ON A.NUMERO = B.NUMERO

GROUP BY A.DATA\_VENDA;

END//

DELIMITER //

CREATE TRIGGER TG\_CALCULA\_FATURAMENTO\_DELETE AFTER DELETE ON ITENS\_NOTAS

FOR EACH ROW BEGIN

DELETE FROM TAB\_FATURAMENTO;

INSERT INTO TAB\_FATURAMENTO

SELECT A.DATA\_VENDA, SUM(B.QUANTIDADE \* B.PRECO) AS TOTAL\_VENDA FROM

NOTAS A INNER JOIN ITENS\_NOTAS B

ON A.NUMERO = B.NUMERO

GROUP BY A.DATA\_VENDA;

END//

18) Vamos incluir novos registros e verificar a tabela consolidada. Para isso digite e execute:

INSERT INTO NOTAS (NUMERO, DATA\_VENDA, CPF, MATRICULA, IMPOSTO)

VALUES ('0107', '2019-05-08', '1471156710' , '235', 0.10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0107', '1000889', 100, 10);

INSERT INTO ITENS\_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)

VALUES ('0107', '1002334', 100, 10);

SELECT \* FROM TAB\_FATURAMENTO;

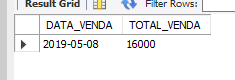
19) E alterar/excluir alguns registros. Para isso digite e execute:

DELETE FROM ITENS\_NOTAS WHERE NUMERO = '0107' AND CODIGO = '1002334';

UPDATE ITENS\_NOTAS SET QUANTIDADE = 400

WHERE NUMERO = '0100' AND CODIGO = '1002334';

SELECT \* FROM TAB\_FATURAMENTO;



# O que aprendemos?

Nesta aula, aprendemos:

* Vimos como funciona campos de auto incremento;
* Aprendemos a determinar valores padrões para os campos;
* Foi mostrado como trabalhar com **TRIGGERs** para executar comandos no momento da inclusão, alteração e exclusão de registros.