



Consolidando o seu conhecimento

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(50));
```

[COPIAR CÓDIGO](#)

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');
```

[COPIAR CÓDIGO](#)

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

```
SELECT * FROM TAB_IDENTITY;
```

[COPIAR CÓDIGO](#)

	ID	DESCRIPTOR
▶	1	CLIENTE1
✱	NULL	NULL

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

```
INSERT INTO TAB_IDENTITY (DESCRIPTOR) VALUES ('CLIENTE2');
```

```
INSERT INTO TAB_IDENTITY (DESCRIPTOR) VALUES ('CLIENTE3');
```

```
INSERT INTO TAB_IDENTITY (ID, DESCRIPTOR) VALUES (NULL, 'CLIENTE4');
```

[COPIAR CÓDIGO](#)

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

```
SELECT * FROM TAB_IDENTITY;
```

[COPIAR CÓDIGO](#)


	ID	DESCRIPTOR
▶	1	CLIENTE1
	2	CLIENTE2
	3	CLIENTE3
	4	CLIENTE4
✱	NULL	NULL

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, 'CLIENTE1');
```

```
SELECT * FROM TAB_IDENTITY;
```

[COPIAR CÓDIGO](#)

	ID	DESCRITOR
▶	1	CLIENTE1
	3	CLIENTE3
	4	CLIENTE4
	5	CLIENTE5
*	NULL	NULL


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;
```

[COPIAR CÓDIGO](#)

	ID	DESCRITOR
▶	1	CLIENTE1
	3	CLIENTE3
	4	CLIENTE4
	100	CLIENTE5
	101	CLIENTE6
*	NULL	NULL

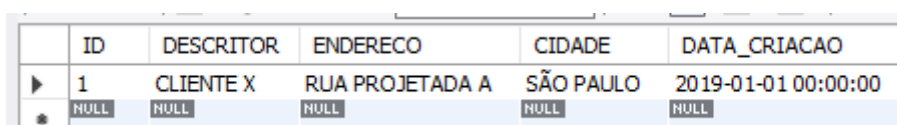
8) Podemos definir padrões para os campos. Com isto um campo pode ter um valor default caso não seja referenciado no comando **INSERT**. Digite e execute:

```
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));
```

[COPIAR CÓDIGO](#)

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;
```

[COPIAR CÓDIGO](#)

	ID	DESCRITOR	ENDereco	CIDADE	DATA_CRIACAO
▶	1	CLIENTE X	RUA PROJETADA A	SÃO PAULO	2019-01-01 00:00:00
*	NULL	NULL	NULL	NULL	NULL

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;
```

[COPIAR CÓDIGO](#)

	ID	DESCRITOR	ENDEREÇO	CIDADE	DATA_CRIACAO
▶	1	CLIENTE X	RUA PROJETADA A	SÃO PAULO	2019-01-01 00:00:00
	2	CLIENTE Y	NULL	Rio de Janeiro	2019-05-14 23:36:56
*	NULL	NULL	NULL	NULL	NULL

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);
```

[COPIAR CÓDIGO](#)

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);
```

[COPIAR CÓDIGO](#)

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

```
INSERT INTO TAB_FATURAMENTO
```

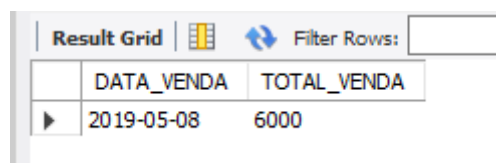
```
SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VEND
```

```
NOTAS A INNER JOIN ITENS_NOTAS B
```

```
ON A.NUMERO = B.NUMERO
```

```
GROUP BY A.DATA_VENDA;
```

```
SELECT * FROM TAB_FATURAMENTO;
```

[COPIAR CÓDIGO](#)

DATA_VENDA	TOTAL_VENDA
2019-05-08	6000

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_VEND
```

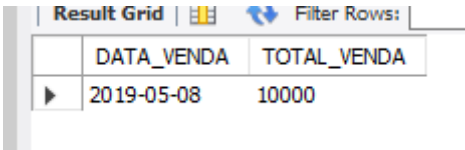
```
NOTAS A INNER JOIN ITENS_NOTAS B
```

```
ON A.NUMERO = B.NUMERO
```

```
GROUP BY A.DATA_VENDA;
```



```
SELECT * FROM TAB_FATURAMENTO;
```

[COPIAR CÓDIGO](#)

	DATA_VENDA	TOTAL_VENDA
	2019-05-08	10000

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 I`  
FOR EACH ROW BEGIN
```

```
DELETE FROM TAB_FATURAMENTO;
```

```
INSERT INTO TAB_FATURAMENTO
```

```
SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VEI  
NOTAS A INNER JOIN ITENS_NOTAS B
```

```
ON A.NUMERO = B.NUMERO
```

```
GROUP BY A.DATA_VENDA;
```

```
END//
```

[COPIAR CÓDIGO](#)

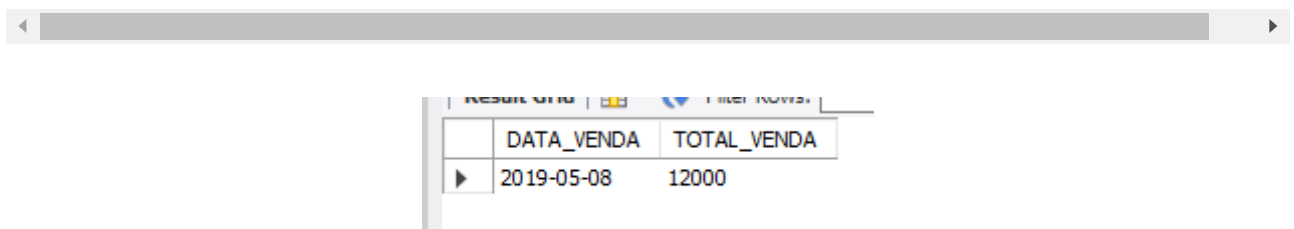
16) Ao inserir novos registros não é mais preciso executar o cálculo da tabela consolidada. Para isso digite e execute:

```
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;
```

[COPIAR CÓDIGO](#)

	DATA_VENDA	TOTAL_VENDA
▶	2019-05-08	12000

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 I
FOR EACH ROW BEGIN
    DELETE FROM TAB_FATURAMENTO;
```

```
INSERT INTO TAB_FATURAMENTO
SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VEI
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 I`  
FOR EACH ROW BEGIN
```

```
DELETE FROM TAB_FATURAMENTO;
```

```
INSERT INTO TAB_FATURAMENTO
```

```
SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VEI
```

```
NOTAS A INNER JOIN ITENS_NOTAS B
```

```
ON A.NUMERO = B.NUMERO
```

```
GROUP BY A.DATA_VENDA;
```

```
END//
```

COPIAR CÓDIGO

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);
```

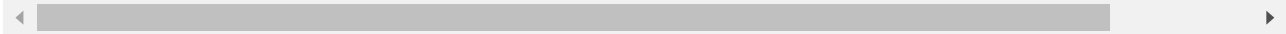
```
INSERT INTO ITENS_NOTAS (NUMERO, CODIGO, QUANTIDADE, PRECO)  
VALUES ('0107', '1002334', 100, 10);
```

```
SELECT * FROM TAB_FATURAMENTO;
```

COPIAR CÓDIGO

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

```
DELETE FROM ITENS_NOTAS WHERE NUMERO = '0107' AND CODIGO = '100  
UPDATE ITENS_NOTAS SET QUANTIDADE = 400  
WHERE NUMERO = '0100' AND CODIGO = '1002334';  
SELECT * FROM TAB_FATURAMENTO;
```

[COPIAR CÓDIGO](#)

Result Grid

	DATA_VENDA	TOTAL_VENDA
▶	2019-05-08	16000