**Manual do iniciante 1.0- MySQL**

**Comandos do banco de dados**

**Comandos citados**

**Nomenclatura criada para teste**

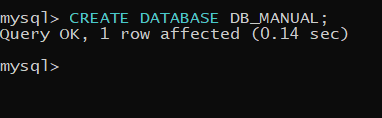
**DATABASE**

**Criando banco de dados**

Após abrir o MySQL setar a senha e LOGIN é necessário criar um **banco de dados** para se trabalhar.

Para criar um banco de dados usar comando

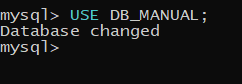
**CREATE DATABASE DB\_MANUAL;**



**Acessando banco de dados**

Ao criar banco de dados usar comando para poder acessar o mesmo.

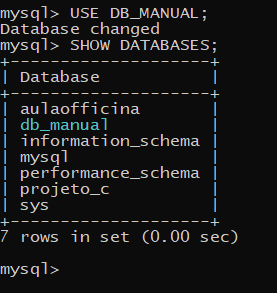
**USE DB\_MANUAL;**

****

**Consultando banco de dados**

Para consultar os bancos de dados criado usar comando

**SHOW DATABASES;**



**Deletando banco de dados**

Caso precise deletar um banco de dados usar comando

**DROP DATABASE BD\_MANUAL;**

**TABLES**

**Criação de tabelas**

Após acessar o banco de dados desejado criar tabelas onde serão armazenados dados usando comando

**CREATE TABLE NOME DA TABELA(**

**NOME COLUNA TIPO (QUATIDADE DE CARACTERES)**

**);**

Os tipos pode ser **“CHAR”** usado para letras, adicionar em colunas onde suas células sempre terá a mesma quantidade de caráter pois usando em células que varei a quantidade de caracteres ocupara espaço desnecessário, **“VARCHAR”** usado para letras adicionar em colunas onde suas células pode variar a quantidade de caráter ocupando espaço apenas do que for preenchido **“INT”** usadopara números com limite máximo de 10 dígitos caso não informe quantidade será usando a quantidade máxima, **“FLOAT”** usado para números fracionados Ex. 100,10 (5,2)**, “ENUM”** usar para campos de multe escolhas onde não pode colocar outro valor a não ser o que foi determinado **Ex. SEXO ENUM (‘M’,’F’);, DATE** usado para datas seguindo a seguinte ordem e formato ‘1995-01-13’ **DATETIME** usado para datas e horas seguindo a seguinte ordem e formato ‘1995-01-13 hh:mm:ss’ **,** as CONSTRAINTS são regras que pode ser vinculado a coluna podendo ser **“PRIMARY KEY”** usado para determina células chaves de cada TUPLAS combinar com **“AUTO-INCREMENT”** para que seja adicionado automático os valores, **“UNIQUE”** usar após determinar tipo e quantidade para determinar que o valor a ser inserido não venha se repetir naquela coluna da tabela , **“NOT NULL”** para determinar que aquela célula não recebera valor nulo.

**EX.**

**CREATE TABLE TB\_CLIENTES (SEXO CHAR (1)); = CHAR**

**CREATE TABLE TB\_CLIENTES (NOME VARCHAR (30)); = VARCHAR**

**CREATE TABLE TB\_CLIENTES (CPF INT); =INT**

**CREATE TABLE TB\_SERVICO (VALOR FLOAT(11,2)); =FLOAT**

**CREATE TABLE TB\_CLIENTES (SEXO ENUM(‘M’,’F’)); = ENUM**

**CREATE TABLE TB\_CLIENTES (COD\_CLIENTE** **INT PRIMARY KEY AUTO\_INCREMENT); = PRIMARY KEY + AUTO\_INCREMENT**

**CREATE TABLE TB\_CLIENTES (CPF INT UNIQUE); = UNIQUE**

**CREATE TABLE TB\_CLIENTES (CPF INT NOT NULL); = NOTNULL**

**CREATE TABLE TB\_CLIENTE**

**(**

**ID\_CLIENTE INT PRIMARY KEY AUTO\_INCREMENT,**

**NOME VARCHAR (30),**

**CPF INT UNIQUE,**

**SEXO ENUM ('M','F'),**

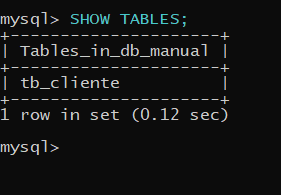
**E\_MAIL VARCHAR (30)**

**);**

**Consultas tabelas**

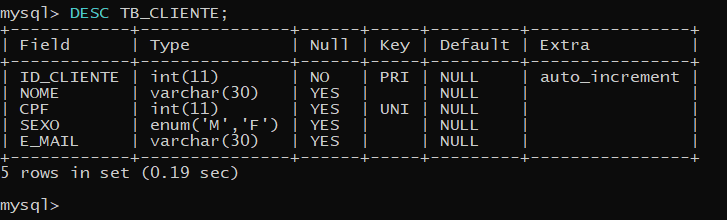
Para visualizar as tabelas que existem dentro Do banco de dados **DATABASE** usa comando

**SHOW TABLES;**

****

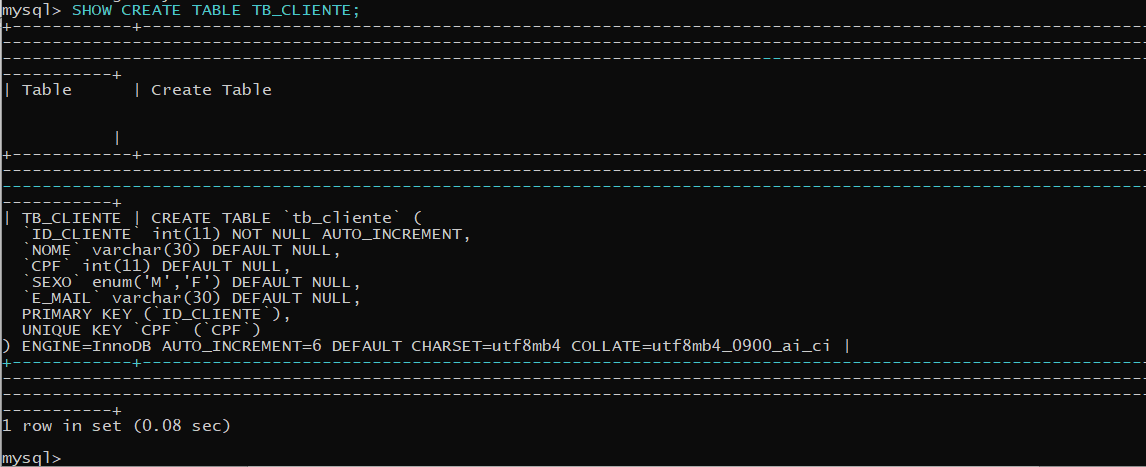
Para visualizar os detalhes da tabela usa comando

**DESC TB\_CLIENTE;**

****

Para visualizar detalhes criação da tabela

**SHOW CREATE TABLE TB\_CLIENTE;**

****

**Adição de dados – INSERT**

**\***sempre usando aspas simples para **CHAR** e **VARCHAR** e **ENUM**

**1º** maneira iserido dados de acordo com a ordem das colunas existente na tabela usando comando

**INSERT INTO TB\_CLIENTE VALUES(**

**NULL,**

**'MARIA AZUL',**

**1234567891,**

**'F',**

**'MARIAZ@AMYSQL.COM.BR'**

**);**

\*em casos onde há auto\_increment na tabela é necessário informa o valor null para que seja auto incrementado ou valor desejado.

**2º** maneira usando comandos onde determina as colunas que iram receber os dados

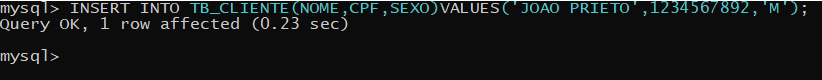
**INSERT INTO TB\_CLIENTE**

**(NOME,CPF,SEXO)**

**VALUES**

**('JOAO PRIETO',1234567892,'M')**

**;**

onde as colunas que não for sitadas receberam os valores nulos nulos “**NULL**”podendo usar ambos comandos para acresentar varias linhas de uma vez só

**INSERT INTO TB\_CLIENTE**

**(NOME,CPF,SEXO)**

**VALUES**

**('JOAO PRIETO',1234567892,'M'),**

**('JOAO BARROSO',1234567829,'M')**

**;**

**INSERT INTO TB\_CLIENTE**

**VALUES**

**(NULL,'JOAO PRIETO',1234567892,'M'),**

**(NULL,'JOAO BARROSO',1234567829,'M')**

**;**

**Alterando dados - UPDATE**

O comando **UPDATE** serve para alterar dados de um celula ou coluna por inteiro caso não seja especificado com **WHERE** a linha o qual deseja fazer a mudanças.

**UPDATE TB\_CLIENTE SET E\_MAIL = 'MARIAAZULA@AMYSQL.COM.BR';**

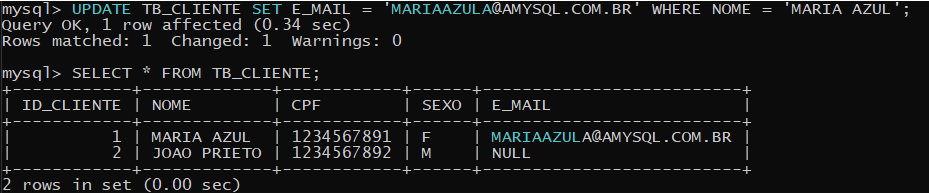
Nesse caso todos as tuplas da coluna e-mail receberia os dados **'MARIAAZULA@AMYSQL.COM.BR'**

**UPDATE TB\_CLIENTE**

**SET E\_MAIL = 'MARIAAZULA@AMYSQL.COM.BR'**

**WHERE NAME = 'MARIA AZUL';**

Nesse caso estou especificando com a clausa **WHERE** que apenas as tuplas onde tenha **NAME** ingual **'MARIA AZUL'** deve ser alterado o e-mail para o valor fornecido.

****

**Deletando dados - DELETE**

Deletando tupla da tabela

para deletar todas as informações existente na tabela usa comando

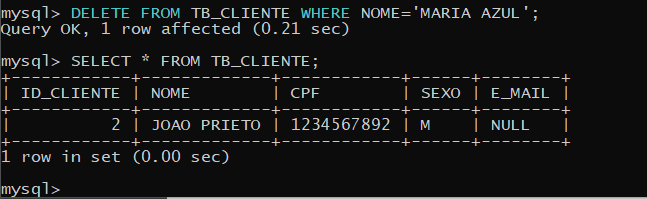
**DELETE FROM TB\_CLIENTE;**

Para deleta linha especifica determina com a causa **WHERE** qual as condições para deletar

**EX.**

**DELETE FROM TB\_CLIENTE**

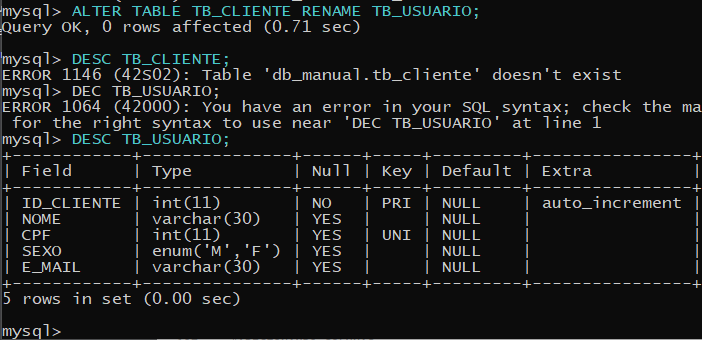
**WHERE NOME='MARIA AZUL';**



**Renomeando tabela**

Para renomear a tabela faz uso do comando **ALTER TABLE** nome da tabela existente **RENAME** e nome desejado.

**ALTER TABLE TB\_CLIENTE RENAME TB\_USUARIO;**



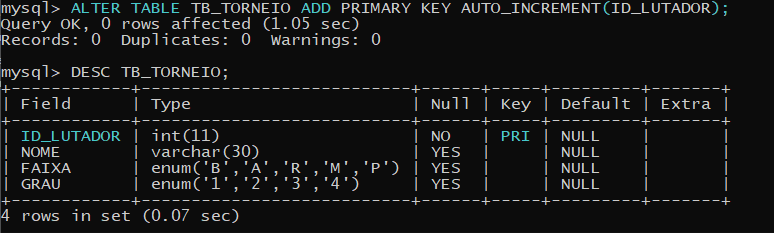
**Incrementado tipo tabela**

Casos precisa determinar que uma coluna seja primary key e não tenha feito isso na criação da tabela faz uso do **ALTER TABLE ADD** onde você determina a coluna.

**EX.** em uma tabela aonde foi determinado que a coluna ID\_LUTADOR seria um INT apenas mas na realidade seria necessário que fosse um PK\_AUTO\_INCREMENTE então faz uso do seguinte comando para incrementar a coluna.

**ALTER TABLE TB\_TORNEIO**

**ADD PRIMARY KEY AUTO\_INCREMENT(ID\_LUTADOR);**

****

**Adicionado coluna na tabela**

Para adicionar coluna na tabela utilizar comando ALTER TABLE ADD seguido do nome da coluna e tipo e suas propiedades desejada.

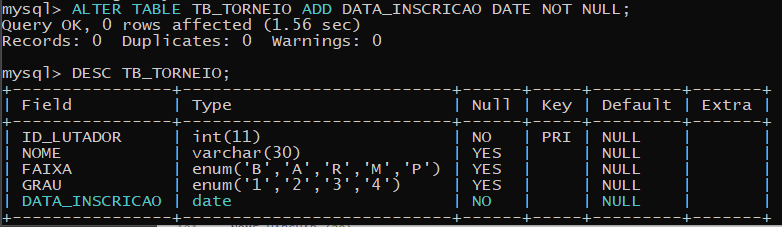
**ALTER TABLE TB\_TORNEIO ADD DATA\_INSCRICAO DATE NOT NULL;**

Caso deseje determinar onde a coluna vai ser criada usar **AFTER** e nome da coluna para criar após.

**ALTER TABLE TB\_TORNEIO**

**ADD DATA\_INSCRICAO DATE NOT NULL**

**AFTER NOME;**

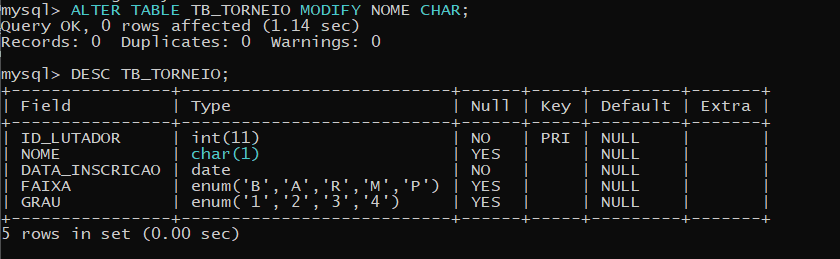
****

**Modificando tipo colunas**

Para modificar colunas usar comando **ALTER TABLE MODIFY** determinado a coluna e o tipo desejado e caso deseje propriedades.

**ALTER TABLE TB\_TORNEIO**

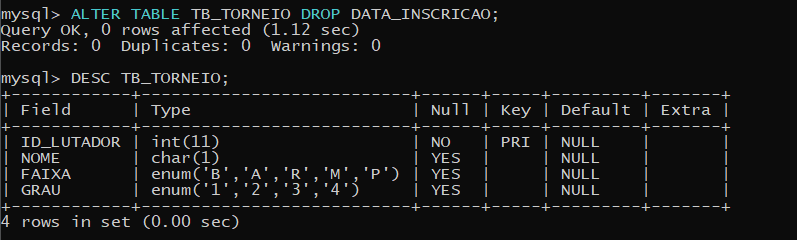
**MODIFY NOME CHAR(30) NOT NULL;**



**Dropando coluna**

Para excluir um coluna utilizar comando **ALTER TABLE DROP** seguido do nome da coluna sem tipo nem propiedades.

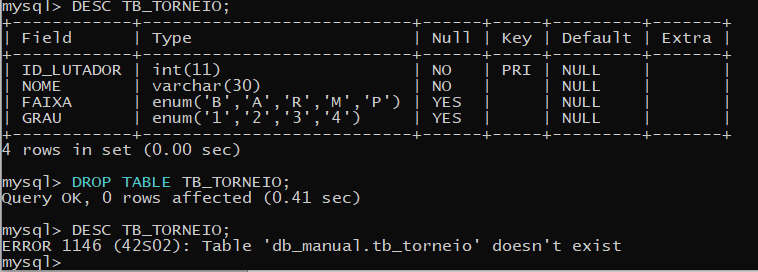
**ALTER TABLE TB\_TORNEIO DROP DATA\_INSCRICAO;**



**Dropando tabela**

Para dropar a tabela fazer uso do comando **DROP TABLE** e nome da tabela.

**DROP TABLE TB\_TORNEIO;**



**Chave estrangeira – FOREIGN KEY**

Para vincular duas tabelas em uma QUERY **SELECT,VIEW,PROCEDURE...** fazendo com que a informação seja coerente é necessário vincular uma chave estrangeira **FOREIGN KEY** em uma das tabelas, normalmente na tabela secundaria a menos importante, exemplo entre duas tabelas um de cadastro de clientes e outra de contato dos clientes a mais importante se torna a do cadastro de cliente logo então a tabela de contato do cliente recebera uma chave estrangeira onde com essa chave identificara o código do cliente na tabela de cadastro de clientes.

**CREATE TABLE TB\_TELEFONE (**

**ID\_TEL INT PRIMARY KEY AUTO\_INCREMENT,**

**DDD INT (3),**

**TELEFONE INT,**

**ID\_CLIENTE\_CE INT NOT NULL,**

**FOREIGN KEY (ID\_CLIENTE\_CE)** #determina qual coluna criada é a chave estrangeira. **REFERENCES TB\_CLIENTE (ID\_CLIENTE))** #determina qual tabela ela vem e qual coluna.

**;**

Nesse caso uso como nome da coluna **FOREIGN KEY** a nome da coluna **PRIMARY KEY** da tabela cliente mais ‘**\_CE’** onde em seguida é defino que essa coluna será uma **FOREIGN KEY** chave estrangeira **REFERENCES** referente a tabela **TB\_CLIENTE** coluna **ID\_CLIENTE** uma **PRIMARY KEY**.

Maneira correta para integração da **FOREIGN KEY** como até mesmo a **PRIMARY KEY** para identificação no dicionário MYSQL é usando o comando

**ALTER TABLE TB\_CARRO**

**ADD**

**CONSTRAINT FK\_TB\_CARRO\_TB\_CLIENTE** #onde aqui determina o “titulo” que ficara no dicionário do MySQL;

**FOREIGN KEY(ID\_CLIENTE\_CE)**

**REFERENCES TB\_CLIENTES(ID\_CLIENTE);**

Aonde **CONSTRAINT** vai atribuir um nome para **FOREIGN KEY** e etc..

Para DROPAR a **CONSTRAINT** usar comando

**ALTER TABLE TB\_CARRO DROP FOREIGN KEY FK\_TB\_CARRO\_TB\_CLIENTE;**

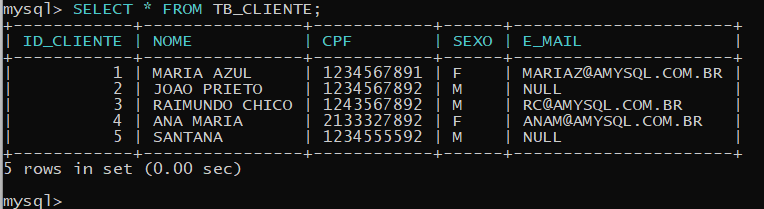
**Projeção de dados**

**Projeção de colunas**

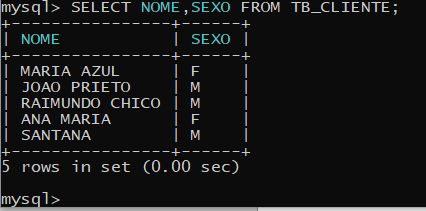
Para visualizar os dados da tabela faz uso da QUERY

**SELECT \* FROM** **TB\_CLIENTE;**

Onde poderá visualizar todos os campos da tabelas e dados cadastrados casos precise ver apenas colunas especificas substituir o asterisco **“\*”** pôr os nome da colunas separados por virgulas.



**SELECT NOME, SEXO FROM** **TB\_CLIENTE;**



**Projeção de tabelas de outro banco de dados**

Para projetar tabela de outro banco ou mencionala em algum código antes de mencionar tabela usa nome do banco e ponto final.

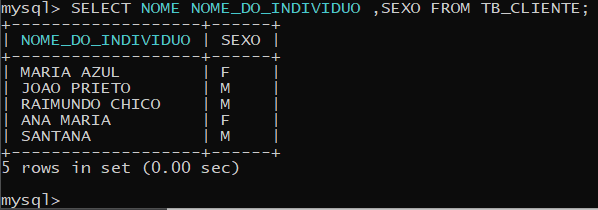
**SELECT \* FROM PROJETO\_OZ.TB\_CLINETES;**

**Renomeando colunas**

Podendo também visualizar o nomes da colunas da forma desejada adicionado moneclatura desejada após os nomes das colunas na QUERY ou da maneira correta adicionado após nome das colunas **AS** **“NOME\_DESEJADO”** que surtira o mesmo efeito.

**SELECT NOME AS “NOME\_DO\_INDIVIDUO”, SEXO FROM** **TB\_CLIENTE;**

**SELECT NOME NOME\_DO\_INDIVIDUO, SEXO FROM** **TB\_CLIENTE;**



**Vinculando tabelas – JOIN**

Para uso adequado da vinculação de tabelas para consultas de dados é recomendado o uso do **JOIN ON** ele é usado para vinculação das tabelas com melhor desempenho.

**#1**

**SELECT C.\*, T.\* FROM TB\_CLIENTE C**

**INNER JOIN TB\_TELEFONE T ON**

**C.ID\_CLIENTE = T.ID\_CLIENTE\_CE;**

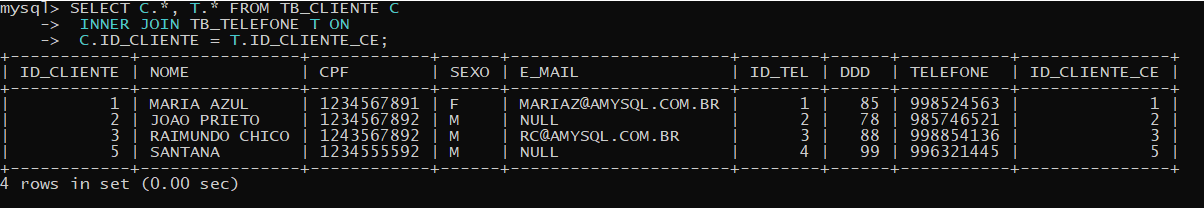
**#2**

**SELECT C.\*, T.\* FROM TB\_CLIENTE C**

**LEFT JOIN TB\_TELEFONE T ON**

**C.ID\_CLIENTE = T.ID\_CLIENTE\_CE;**

#1-Nesse caso foi determinado que a **TB\_CLIENTE** com a apelido de **C** buscasse todos os clientes no qual o **CD\_CLIENTE** fosse igual **CD\_CLIENTE\_CE** da **TB\_TELEFONE** apelidada de **T** e projetasse as duas tabelas **(C.\*, T.\*)** as informações ao qual fosse coerente.

****

#2-Nesse caso foi determinado que a **TB\_CLIENTE** com a apelido de **C** buscasse todos os clientes no qual o **CD\_CLIENTE** fosse igual **CD\_CLIENTE\_CE** ou nulo na **CD\_CLIENTE\_CE** da **TB\_TELEFONE** apelidada de **T** e projetasse as duas tabelas **(C.\*, T.\*)** as informações ao qual fosse coerente.

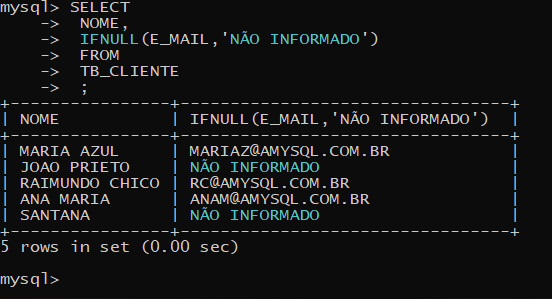
**IFNULL**

Função IFNULL usado para se o valor for nulo não trazer NULL exemplo de uso

****

**SELECT NOME, IFNULL (E\_MAIL,’NÃO INFORMADO’)**

**FROM TB\_CLIENTE;**

****

**VIEW**

**Criação de VIEW**

As VIEWS são criadas a partir QUERY de projeção

**CREATE VIEW VW\_NULOS AS** em seguida QUERY SELECT desejada

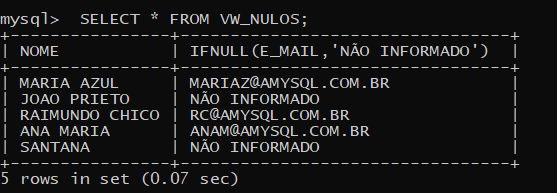
**EX.**

**CREATE VIEW VW\_NULOS AS**

**SELECT NOME, IFNULL(E\_MAIL,'NÃO INFORMADO') FROM TB\_CLIENTE ;**

**Consulta VIEW**

**As VIEWS são visualizadas como tabelas com os comandos SHOW TABLES ou SELECT \* FROM** **VW\_NULOS;**

****

**PROCEDURE**

**Criação de procedure**

Para criação de **PROCEDURE** determine um delimitador diferente de “**;** ” para que assim a **PROCEDURE** seja executada até final.

**EX.**

**DELEMITER** **#**

Para verificar se delimitador mudou verificar com comando **STATUS**

Após determina o delimitar usa comando para criação de procedure

**EX.**

**CREATE PROCEDURE VALOR (VALOR1 INT, VALOR2 INT)**

**BEGIN**

**SELECT (VALOR1 + VALOR2) AS “SOMA”**

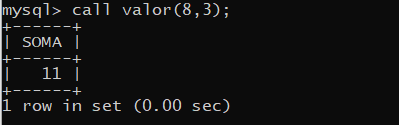
**END**

**#**

**Aderindo valores a procedure**

Para chama a procedure usar comando **CALL** determinando o valor que irá preencher as variáveis no caso 8 e 3

**CALL VALOR (8,3)**



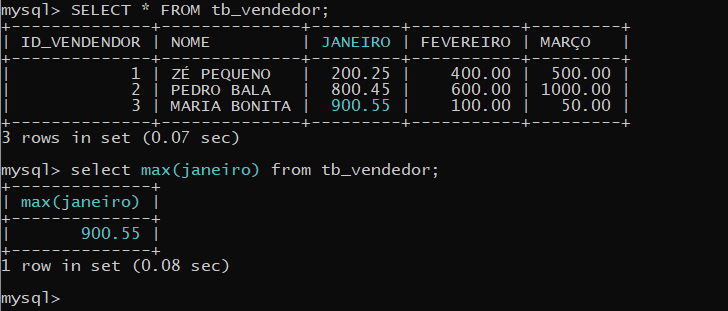
**FUNÇÕES**

**Máximo**

**MAX** comando de projeção usado para determinar o valor máximo atingido por aquela coluna

**EX.**

**SELECT MAX(JANEIRO) FROM TB\_VENDEDOR;**

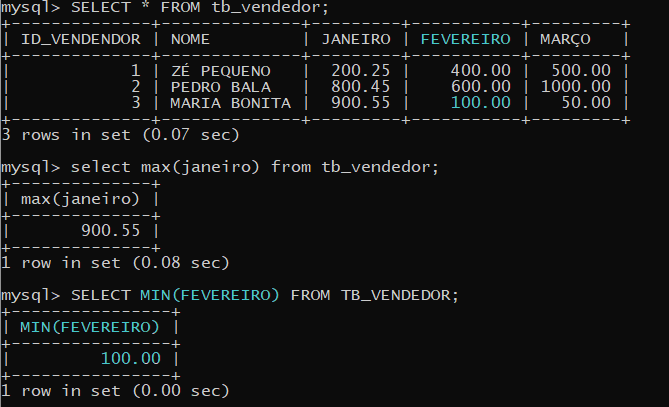
****

**Mínimo**

**MIN** comando de projeção usado para determinar o valor mínimo atingido por aquela coluna

**EX.**

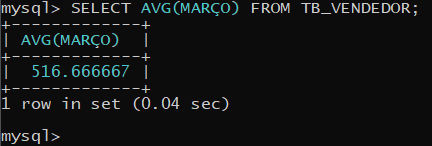
**SELECT MIN(FEVEREIRO) FROM TB\_VENDEDOR;**

****

**Média**

**AVG** comando de projeção usado para determinar o valor da média atingido por aquela coluna somando todos os valores encontrado e dividindo pela quantidade.

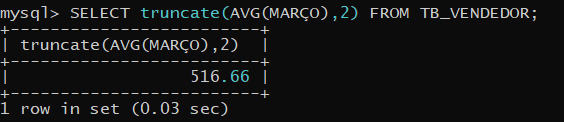
**SELECT AVG(MARÇO) FROM TB\_VENDEDOR;**

****

**Truncar**

**TRUNCATE** comando de projeção usado para determinar a quantidade de casas após as virgula usando “virgula e número de casas desejadas” após determinar a coluna.

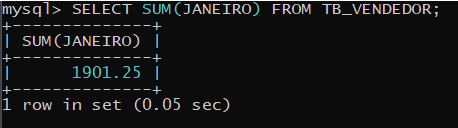
**SELECT TRUCANTE(AVG(MARÇO) ,2) FROM TB\_VENDEDOR;**

****

**Soma**

**SUM** comando de projeção usado para soma valores total da coluna.

**SELECT SUM(JANEIRO) FROM TB\_VENDEDOR;**

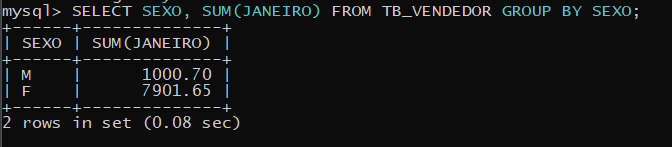
****

**Agrupar**

**GROUP BY** comando para agrupar podendo usar junto com função **SUM**

**SELECT SEXO, SUM(JANEIRO) FROM TB\_VENDEDOR GROUP BY SEXO;**

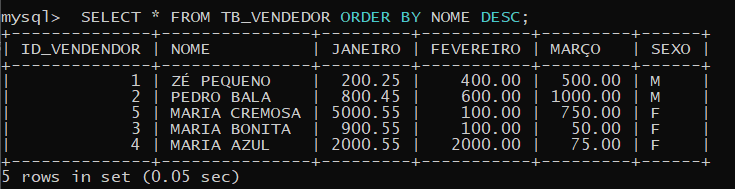
Nesse caso determinei que ele me projetasse o valor total de venda no mês de janeiro mas me agrupasse por sexo.

****

**Ordenar**

**ORDER BY “ASC, DESC”** serve para ordenar por determinada coluna podendo ordenar por ordem decrescente “**DESC**” e crescente “**ASC**”.

**SELECT \* FROM TB\_VENDEDOR ORDER BY NOME DESC;**

****

**SUBQUERY**

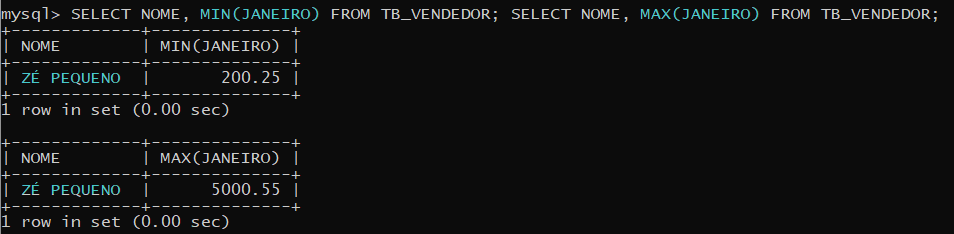
SUBQUERY são usados no **WHERE** para tipos de função onde não são permitidas usar em combinações

**EX.**

Você deseja encontra o nome e o valor máximo do vendedor que vendeu menos em janeiro usando a QUERY

**SELECT NOME, MAX(JANEIRO) FROM TB\_VENDEDOR;**

O resultado será um valor incorreto um valor incorreto



Nessas duas QUERY onde pedir o mínimo e o máximo ele determinou o mesmo vendedor,

Para projetar o resultado correto usa as SUBQUERY

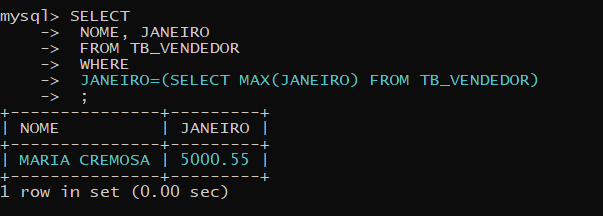
**SELECT**

**NOME, JANEIRO**

**FROM TB\_VENDEDOR WHERE**

**JANEIRO = (SELECT MIN(JANEIRO) FROM TB\_VENDEDOR)**

**;**

****

**TRIGGERS**

Triggers são gatilhos de automação usados quando há alterações nas tabelas e gravando as alterações em uma tabela auxiliar de auditoria.

Comando de Ex.

**DELIMITER !** #comando para mudar delimitador

**CREATE TRIGGER BK\_CLIENTES #**1

**AFTER INSERT ON TB\_CLIENTES** #2

**FOR EACH ROW**

**BEGIN**

**INSERT INTO BK\_TB\_CLIENTES VALUES**

**(NULL,**

**NEW.ID\_CLIENTE,** #3

**NEW.NOME,**

**NEW.SEXO,**

**NEW.IDADE,**

**NEW.CLIENTE\_DESD,**

**NEW.EMAIL,**

**'I',** #4

**NOW(),** #5

**CURRENT\_USER));** #6

**END!**

**DELIMITER ;**

**#1**-comando de criação de trigger onde nomear a trigger.

**#2**-comando **AFTER**(pós) ou **BEFORE**(antes) definir que momento a trigger será executada tipo **“INSERT,UPDATE,DELETE”** que tipo de alteração ira ativa-la e qual tabela ira ser monitorada.

**#3-NEW.** Ou **OLD.** Comando usado para definir em que momento os dados serão coletados antes da alteração **OLD.** Ou após **NEW.**

**INSERT = NEW**

**DELETE = OLD**

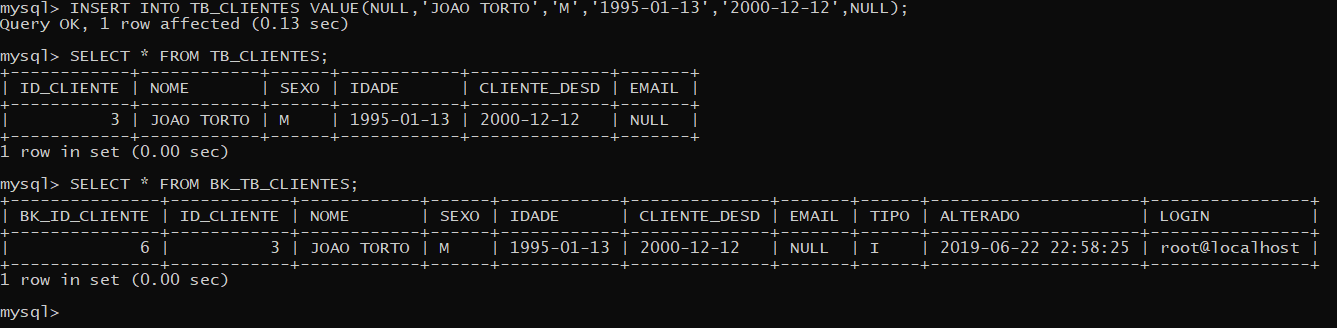
**UPDATE = NEW** ou **OLD.**

**#4-ENUM** adicionado para determinar o tipo de alteração em cada trigger.

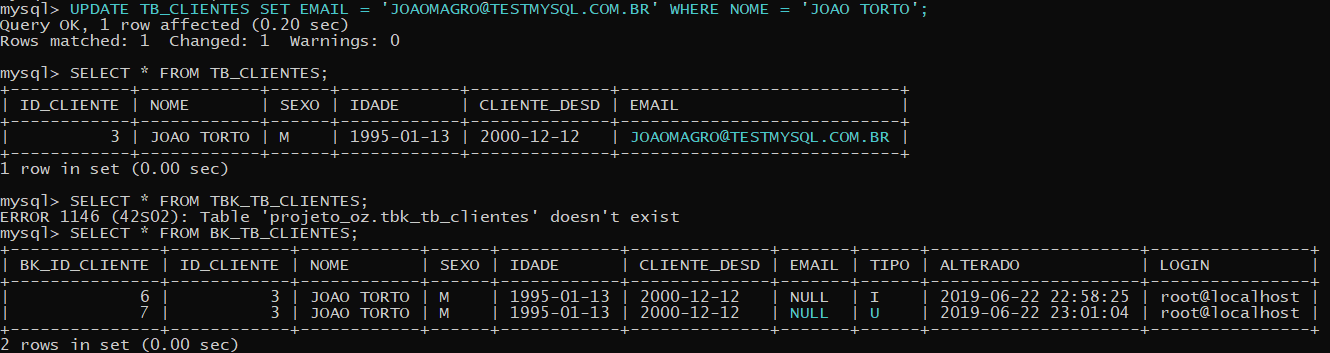
**#5**-registrar a hora da alteração

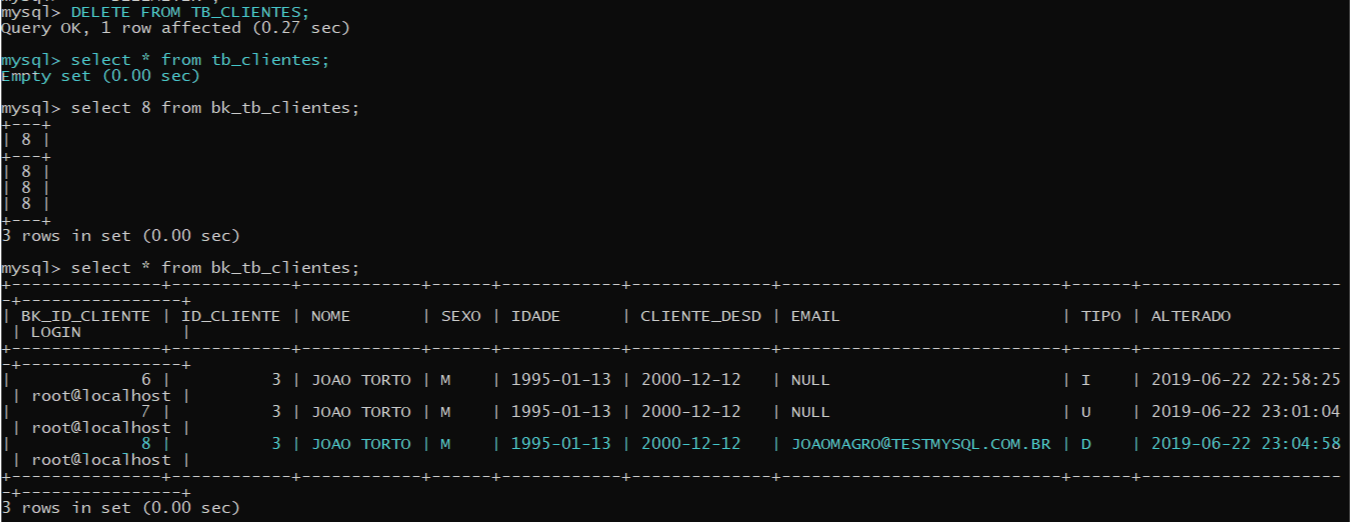
**#6**-registra o usuário.

**INSERT**

****

**UPDATE-**Registrado UPDATE como OLD. pegando assim os valores anterior.

****

**DELETE**

**CURSORES**

**CURSOR** serve para armazenar e manipular dados na memória da maneira desejada eles são executados dentro das **PROCEDURES**

**EX.**

**DELIMITER !**

**CREATE PROCEDURE VENDA\_T()**

**BEGIN**

**DECLARE FINAL INT DEFAULT 0;** #1

**DECLARE VEND INT**

**DECLARE VJAN, VFEV, VMAR, TOTAL, MEDIA FLOAT (10,2);** #2

**DECLARE AUTO CURSOR FOR(**

**SELECT CD\_VENDEDOR\_FK,JAN,FEV,MAR FROM TB\_VENDAS\_TRI);** #3

**DECLARE CONTINUE HANDLER FOR NOT FOUND SET FINAL = 1;** #4

**OPEN AUTO;** #5

**REPEAT** #6

**FETCH AUTO INTO VEND,VJAN,VFEV,VMAR;** #7

**IF NOT FINAL THEN** #8

**SET TOTAL = VJAN+VFEV+VMAR;** #9

**SET MEDIA = TOTAL/3;**

**INSERT INTO TB\_VENDAS\_T VALUES(VEND,VJAN,VFEV,VMAR,TOTAL,MEDIA);**

**END IF;** #8

**UNTIL FINAL END REPEAT;** #6

**CLOSE AUTO;** #5

**END**

**!**

**DELIMITER ;**

**#1-**declarando variável contadora onde vai determinar quando o loop vai parar determinar valor “0”**.**

**#2-**declaração do tipo de variável é feito no final de declarar todas variáveis.

**#3-**declarado o cursor e quando colunas e tabela ele vai lê os dados linha a linha, a query interna não deve ter ponto e vírgula.

**#4**-declarando controlador onde NOT FOUND definir quando não encontra dados SET na variável FINAL o valor 1.

**#5**-ultilizado para usar o cursor começa a trabalhar com ele na memória RAM e finalizar.

**#6**-serve para repetir o comando até final do cursor.

#7-buscar para dentro do cursor as variáveis para receber os valores de acordo com a query #3 definida.

#8-enquanto a variável FINAL não receber valor 1 continuar loop.

#9-comando selecionado a ser executado.

**XAMPP**