https://www.youtube.com/watch?v=WB4ykHGr7w8&list=PLucm8g\_ezqNoAkYKXN\_zWupyH6hQCAwxY&index=30

Banco de dados/impala

Livro Introdução a linguagem SQL

Banco de dados leves são banco de dados pequenos e de poucos registros, onde pode ser mandado o arquivo para outras pessoas

Banco de dados centralizados é um banco de dados com bastante usuários aonde precisa de um servidor para hospedar e manipular inúmeros usuários ao mesmo tempo

Impala Comandos:

Sempre termine com ;

DESCRIBE: Mostra as informações da tabela.

CREATE TABLE nome da tabela as SELECT \* FROM nome da tabela que vai criar

SELECT \* FROM xxx LIMIT numero de linhas que o código vai pegar

INSERT/SELECT \* FROM xxx: Serve para selecionar uma tabela

CREATE TABLE xxx(xxx tipo(string ou float etc) , xxxk, xxx, xxx, xxx, xxx): Serve para criar uma nova tabela com seus parâmetros

INSERT INTO nome da tabela(nome das coluna) VALUES(informações da tabela): Serve para inserir valores em uma tabela.

SELECT count(nome), nome FROM cidades GROUP BY nome HAVING COUNT(nome) > 1 ORDER BY count(nome) DESC;: serve para contar quantas vezes apareceu aquele nome na coluna e mostrar em ordem decrescente a quantidade

select nome da coluna,count(\*)

from nome da tabela

group by nome da coluna

having count(\*) > 1;

: para saber se alguma informação se repete na coluna

Alter table nome da tabela change nome da coluna que vai ser alterada nome da nova coluna tipo de dado que vai ser desta coluna : serve para alterar o nome de uma coluna

SQL COMANDOS:

TIPOS PRIMITIVOS BANCO DE DADOS:

NÚMERO:

INTEIRO:

TINYINT — número inteiro muito pequeno (tiny);

SMALLINT — número inteiro pequeno;

MEDIUMINT — número inteiro de tamanho médio;

INT — número inteiro de tamanho comum;

BIGINT — número inteiro de tamanho grande;

REAL:

DECIMAL — número decimal, de ponto fixo;

FLOAT — número de ponto flutuante de precisão simples (32 bits);

DOUBLE — número de ponto flutuante de precisão dupla (64 bits);

Logico:

BIT — um campo de um bit.

Data/tempo:

DATE — o valor referente a uma data no formato 'CCYY-MM-DD'. Por exemplo 1985-11-25 (ano-mês-dia). O 'CC' se refere aos dois dígitos do século (Century, em inglês);

TIME — um valor horário no formato 'hh:mm:ss' (hora:minutos:segundos);

TIMESTAMP/ DATETIME — timestamp é uma sequência de caracteres ou informação codificada que identifica uma marca temporal ou um dado momento em que um evento ocorreu. No MySQL, ele tem o formato 'CCYY-MM-DD hh:mm:ss' – neste caso, seguem a padronização ISO 8601;

YEAR — armazena um ano no formato 'CCYY' ou 'YY';

Literários:

CHAR — uma cadeia de caracteres (string), de tamanho fixo e não-binária;

VARCHAR — uma string de tamanho variável e não-binária;

BINARY — uma string binária de tamanho fixo;

VARBINARY — uma string binária de tamanho variável;

BLOB — um BLOB (Binary Large OBject – OBjeto Grande Binário) pequeno;

TINYBLOB — um BLOB muito pequeno;

ENUM( , ) serve para colocar um caractere especifico

MEDIUMBLOB — um BLOB de tamanho médio;

LONGBLOB — um BLOB grande;

TINYTEXT — uma string não-binária e de tamanho bem reduzido;

TEXT — uma string não-binária e pequena;

MEDIUMTEXT — uma string de tamanho comum e não-binária;

LONGTEXT — uma string não-binária de tamanho grande;

ENUM — de acordo com o manual do MySQL, é uma string, com um valor que precisa ser selecionado de uma lista predefinida na criação da tabela;

SET — é um objeto que pode ter zero ou mais valores – cada um dos quais precisa ser escolhido de uma lista de valores predeterminados quando da criação da tabela.

Constranges( condições de variáveis)

NOT NULL serve para obrigar aquela variável ter um valor

Default serve para se estiver vazio ela vai se preencher com algum valor em especifico.

ex qtd smallint default 0

AUTO\_INCREMENT serve para incrementar por ordem com números

UNIQUE :Faz com que não repita a mesma informação em uma variável

UNSIGNED: faz com que os numeros não tenham sinal como de negativo

DEFAULT ‘XXX’: Serve para se não registar algo na variável ela vai completar com o a palavra escrita nos xxx

PRIMARY KEY (nome da coluna que vai ser a chave primaria)Precisa criar uma chave primaria, que é aquela que não vai se repetir.

CREATE DATABASE nome do banco de dados; : Serve para criar um novo banco de dados.

default character set utf8

default collate utf8\_general\_ci

: serve para criar a tabela já com a configuração padrão de caracteres para brasil.

DROP DATABASE nome do banco de dados; : serve para apagar o banco de dados

ALTER TABLE nome da tabela

Drop table if exists nome da tabela

drop table nome da tabela

vai apagar a tabela inteira permanentemente CUIDADO!!!!

PARA RESTAURAR

Para zer a restauração isso sendo no pgadmin

DROP COLUMN nome da coluna

Serve para apagar uma coluna da tabela em especifico.

USE nome do bando de dados: serve para abrir o banco de dados

CREATE TABLE (if not exist) se a tabela n existir

nome da tabela ( nome das colunas e precisa colocar o tipo de dado que a coluna vai ser,xxx,xxx,xxx ; ) Serve para descrever o nome das colunas e o tipo dela

Ex :

create table pessoas (

id int not null auto\_increment,

nome varchar(30) not null,

nascimento date,

sexo enum ('M' , 'F'),

peso decimal(5,2),

altura decimal (3,2),

nascionalidade varchar(20) default 'Brasil',

primary key(id)

) default charset = utf8;

INSERT INTO tabela

(nome das colunas separadas com virgula) se você colocar as informações das colunas já na ordem que elas estão então não precisa colocar o nome das colunas antes

VALUES

(valores das colunas precisa usar as ‘ xxx ‘ ) , pode colocar quantas informações quiser, mas precisa ser colocado por parênteses.(xxx)

Ex :

insert into pessoas values

(default, 'Maria', '1994-10-05', 'F', '65.5', '1.65', 'Brasil'),

(default, 'Lucas', '2001-10-02', 'M', '90', '1.93', default),

(default, 'Tainara', '2002-09,04', 'F', '76.5', '1.69',default);

ALTER TABLE nome da tabela

ADD COLUM nome da coluna e o tipo da coluna (se não colocar nada a coluna vai para o final da fila); AFTER nome da coluna que vai vir antes dessa ou seja a coluna vai pra depois dessa ; se quiser colocar a coluna em algum lugar em especifico; FIRST se quiser que a coluna seja a primeira.

Serve para adicionar outra coluna no seu banco de dados

Alter table nome da tabela

MODIFY COLUMN nome da coluna e o tipo serve para mudar o tipo da coluna seja como tamanho

Alter table nome da tabela

CHANGE COLUMN nome da coluna antiga nome da coluna nova tipo da coluna

Serve para mudar o nome da coluna em uma tabela

Alter table nome da tabela

Add primary key( nome da coluna que vai ser sua chave primaria), serve para declarar uma coluna que vai ser a chave primaria caso não tenha colocado no inicio

ALTER TABLE nome da tabela

RENAME TO nome da nova tabela

Serve para mudar o nome da TABELA

Comando SELECT

Select \* from xxx(nome da tabela): seleciona todos as colunas e todas as linhas.

Voce tambem pode acessar colunas em especifico

SELECT xxx,xxx,xxx ( nome das colunas)

FROM xxx ( nome da tabela).

no comando select podemos utilizar para fazer calculos e mostrar na tabela por exemplo

Select Produto\_id, descrição, price, price \* 1.07 AS taxa\_price

FROM poduto,

Este codigo vai pegar a coluna price e calcular ela mais 7% e ainda criar outra tabela mostrando essa taxa de price.

AS o comando as significa alias aonde voce pode da um gatilho para qualquer coisa por exemplo

Select price as untaxed\_price,

ela não vai MUDAR o nome na tabela, mas na hora de DEMONSTRAR vai aparecer o nome em que você escolher.

O comando ROUND(xxx,xxx) serve para arredondar numeros decimais, voce para o primeiro parametro e no segundo voce coloca em quantas casas decimais você quer que apareça o resultado ex:

Selecgt product\_id, description, price, round(PRICE \* 1.07,2) AS TAXED\_PRICE

from product

neste comando na coluna price ele vai pegar 7% do valor e somar e mostrar com outro nome, e ainda por cima os valores estarão com apenas duas casas decimais.

Conectores, voce pode fazer conectores de texto no select utilizando um pipe duplo (||)

ex:

Select nome,

cidade || ‘ , ‘ || estado as localização

from customer

nome

localização

mma

brasilia , DF

mri

rio de janeiro, RJ

pode fazer conectivos infinitos, e pode se concatenar com qualquer tipo de dado: numero,datas,texto etc …

ou utilizando o concat, que se usa

SELECT CONCAT (‘O livro’, nome\_livro,’ custa ’, preco\_livro,’ reais’) AS livro

FROM tbl\_livros

WHERE id\_livro = 3;

Order by xxx; : mostra ele ordenado por nome. (se adiciondar um DESC no final ele ordena de baixo para cima

Select xxx, xxx, xxx from xxx;

Orderby xxx: vai mostrar as colunas em especifico de uma tabela especifico e as colunas vão aparecer na ordem que você escreveu

Where

o Where é uma instrução para filtrar os dados que estamos procurando

ex: select \* from station\_data

where year = 2010

aqui vai pegar tudo da tabela station\_data aonde o ano seja igual a 2010.

Se voce quiser no where não selecionar algo pode se usar o != ou o <> ex

select \* from station

where year != 2010

ou

select \* from station

where year <> 0

neste codigo ele vai selecionar tudo menos aonde na coluna ano seja igual a 2010

Select \* from xxx

Where xxx = xxx

Order by xxx; Seleciona uma linha em especifico ou linhas que você quer

Ex:Select nome, descrição, carga FROM cursos

WHERE ano = ‘2016’

Order by nome; neste código ele vai selecionar a coluna (nome,descrição e carga) da tabela cursos aonde o ano é igual a 2016 e vai ordenar por nome.

voce pode identificar os valores nulos, escrevendo NULLS first ou last isso organiza os valores nulos ou para aparecer primeiro ou no final

ex:

select \* from preco order by nome\_produto nulls first

voce pode ordenar mais de uma coluna essa segunda coluna pega prioridade CASO a primeira coluna tenha um código repetido

voce pode usar dcs para ordem decrescente

Where xxx in (xxx,xxx); ele pega a seleção em especifico, ex

Where ano in (2014, 2016); pega o ano que esta em 2014 e 2016.

Select \* from xxx

Where xxx BETWEEN xx AND xxx ela seleciona alguma coisa entre a outra ex:

Where totaulas BETWEEN ‘20’ AND ‘30’ seleciona o total de aulas entre 20 e 30

tambem pode se usar o maior igual e menor igual com and que faz a mesma função do between

ex: Select \* from state

where year >= 2005 and year <2010

vai pegar entre o ano de 2005 e 2010.

se quiser selecionar uma coisa OU outra basta utilizar o operador OR ex

select \* from state

where month = 3 or month = 6 or month = 9 or month 12

ou nessa mesma instrução pode se usar o operador IN que vai fornecer uma lista de valores.

select \* from state

where month IN (3,6,9,12).

e se quiser valores exceto esses basta colocar o NOT antes.[

para usar um where com texto precsa usar aspas simples ‘ xxx ’ para não confundir com coluna.

a instrução length() serve para contar o numero de caracteres de um valor especifico(Normalmente String), usado para verificar por exemplo se todos os códigos tem todos os números

ex: select \* from xxx

where length(xxx) != 6, vai mostrar todos os codigos aonde o numero dele é diferente de 6.

Operador LIKE

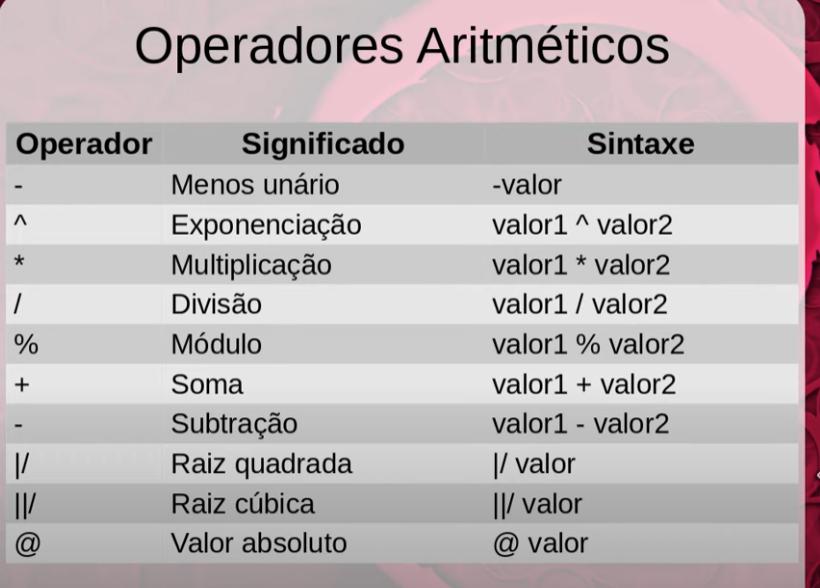
Select \* from cursos

Where nome like ‘p%’;

Ele mostra todos os cursos que começa com a letra P

O like serve para buscar algo em especifico a palavra like significa “parecido” e precisa colocar o %.

Se você quiser achar nomes que tenham o A em qualquer lugar basta colocar a porcentagem no inicio e fim “ %a%” e se quiser palavras que não tenha o A basta colocar o “not” antes do like.



Quando se usa “\_” e porque necessariamente precisa ter algo depois daquilo

o valor null e um valor em branco na tabela, para identificar valores nulos precisa usar o is null ou is not null

ex: select \* from xxx

where xxx is null;

para buscar os valores ou inclui los no select voce precisa indicar um nulo

para transformar nulos em alguma coluna por um valor em específico basta usar o coalesce()

ele transforma valores possivelmente nulos em algo que voce queira.

ex: select \* from xxx

where coalescre(xxx,0) <=0.5) aqui ele vai pegar a coluna e aonde tiver nullo ele vai subsituir por 0. e selecionar aonde e menor ou igual a 0.5

Distinct : serve para não mostrar palavras repetidas, ele considera uma ocorrência dentro do local

Ex

Select distinct nacionalidade from gafanhotos;

Vai mostrar os nomes da coluna que apareceram mas não vai repetir.

Função agregação

ele retorna apenas 1 valor

Serve para selecionar ou totalizar algo

Ex

Select count(\*) from cursos;

COUNT serve para contar algo

Vai contar os cursos os itens da lista

Mostrar a quantidade

voce pode usar numeros no goupby para mostrar, se não quiser digitar o nome das colunas

ex: Select ano, mes, count(\*) as contagem from xxx

where tornado = 1

goupb by 1, 2

esse codigo vai selecionar o ano e mes, contar as linhas onde vai ter tornado e mostrar as colunas agrupadas.

“max” ele mostra o máximo de uma coluna

Ex

Select max(carga) from cursos Where ano = ‘2016’;

Vai mostrar o total de carga no ano de 2016

O “min” mostra o mínimo

O “sum” mostra a soma do que foi selecionado

O “avg” ele tira a média da coluna que você quiser.

GROUP BY

Agrupamento

Serve para agrupar um determinado tipo de valor e vai mostrar a quantidade de vezes que aquele valor apareceu

Ex:

Select carga, count(\*) From cursos

GROUP BY carga

Aqui vai agrupar cargas que se repetem e mostrar a quantidade

Having ele serve para especificar algo que você queira ele filtra agregação.

não se pode fazer buscas com where em campos agregados ex

Select year, SUM(precipitation) as total\_precipitation

from station\_data

WHERE total\_precipitation > 30

group by year

este código não vai funcionar

precisa adicionar o having e tirar o where

ex Select year, SUM(precipitation) as total\_precipitation

from station\_data

group by year

having total\_precipitation > 30

se utilizar o racle basta colocar having agregação(coluna) e a condição

Ex

Select ano, count(\*) from cursos

Group by ano

Having count(ano) >= 5

Order by count(\*);

Esse código vai mostrar apenas quem tem o ano maior ou igual a 5.

O having so funciona com o campo que esteja no group by

CASE

serve para mapear uma ou mais condições para um valor correspondente a cada condição.

a sintaxe é a seleção xxx when condição then valor e termina com end.

ex:Select report\_code, year, month, day, wind\_speed,

CASE

WHEN wind\_speed >= 40 THEN ‘HIGHT’

WHEN wind\_speed >= 30 THEN ‘MODERATE’

ELSE ‘LOW’

end as wind\_severity

from station\_data

neste comando ele vai pegar a coluna wind\_speed e aonde tiver maior ou igual a 40 ele vai escrever HIGHT, aonde estiver entre 30 e 40 vai escrever MODERATE e o restante é LOW

Manipulando linhas

UPDATE nome da tabela

SET nome da coluna = ‘ qual vai ser o que vai ser alterado‘

WHERE nome da coluna da linha que vai ser identificado = ‘ a referencia para achar a linha ‘; ( normalmente colocamos a chave primaria aonde não se repete informações) , se quiser mudar mais de uma informação na linha basta colocar virgula no SET.

Serve para mudar valores em uma linha

voce pode fazer calculos no set

outro exemplo utilizando upper() que serve para deixar as letras maiusculas.

UPDATE nome\_da\_tabela SET email = UPPER(EMAIL)

aqui deixa todos os nomes da tabela da coluna email em maiusculo.[

pode se modificar varias informações apenas utilizando a virgula apos o SET

ex: UPDATE nome\_da tabela SET first\_name = upper(first\_name),

last\_name = upper(last\_name).

vai colocar as informaçoes das colunas em maiusculos.

DELETE FROM nome da tabela

WHERE nome da linha = ‘ a informação que esta na célula ‘; Serve para apagar uma linha da tabela

a recomendado e utilizar um select \* para ter certeza de qual tabela ela sera excluída.

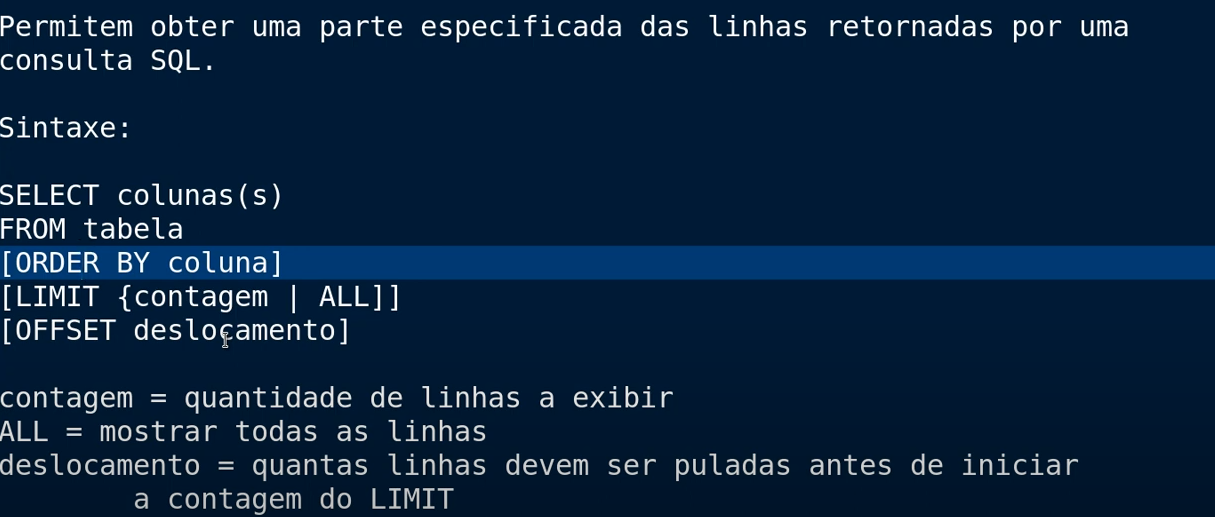
tem que tomar cuidado com o delete pois pode apagar todas as informações da tabela

LIMIT numero; ele limita em quantas vezes haverá a alteração de uma informação por exemplo o numero de linhas limit 4;

offset: permite obter uma parte especificada das linhas retornadas pela consulta, o limit tambem faz

o offset limita onde ele vai começar , me mostre depois da 3 linha

essas duas de cima não podem ser usadas sem o orderby



TRUNCATE nome da tabela; serve para apagar TODAS as informações de uma tabela.não apaga a tabela, apenas os registros

TRUNCATE TABLE nome\_tabela;

UPDATE nome\_tabela

SET CAMPO = "novo\_valor"

WHERE CONDIÇÃO

onde:

Nome\_tabela: nome da tabela que será modificada

Campo: campo que terá seu valor alterado

Novo\_valor: valor que substituirá o antigo dado cadastrado em campo

Where: Se não for informado, a tabela intera será atualizada

Condição: regra que impõe condição para execução do comando

Colocando uma chave estrangeira

Coloca

ADD FOREIGN KEY (nome da coluna que vai ser a chave estrangeira) REFERENCES tabela da entidade(a chave primaria dessa tabela)

OU

ALTER TABLE ‘nome\_do\_banco\_de\_dados’.’nome\_da\_tabela’

ADD CONSTRAINT ‘nome\_para a chave estrangueira’

FOREIGN KEY (cave estrangeira)

REFERENCES nome do banco de dados.nome da segunda tabela(chave primaria da tabela 2)

ou para CRIAR tabela, ja com chave estrangeira basta colocar o nome da coluna igual a da tabela principal ex.

cod\_cliente INT NOT NULL REFERENCES clientes(cod\_cliente);

a parte do inicio precisa ser exatamente igual a chave primaria

Ver informações de uma tabela como tipo

JOIN

Inner join

ele serve para mesclar duas tabelas, mas para isso precisa definir um ou mais atributos comum entre elas para que os registros se alinhem

Ex select gafanhotos.nome, cursos.nome, cursos.ano

From gafanhotos inner join cursos

On cursos.idcurso = gafanhotos.cursopreferido

Order by gafanhotos.nome

Esse serve para juntar o curso com gafanhotos apenas que tenham valores relacionados

ele não vai mostrar informações se esses valores não estiverem relacionados com a outra tabela.

outro exemplo:

SELECT Order\_if, Customer.customer\_id,

order\_date

from customer INNER JOIN customer\_order

on customer.customer\_id = customer\_order.customer\_id

aqui ele vai pegar as colunas fazer o inner join e igualar as duas colunas que estão relacionadas

left join

vai mesclar as informações mas com um adicional de que todos os registros da tabela quem vem a ESQUERDA(LEFT) vai ser incluidos, mesmo se não tiverem registros filhos na tabela da esquerda.mas os registros que eles não possuem apareceram como nullo

Pode colocar o left ou right outer join vai juntas as tabelas que não precisam estar com o mesmo tanto de nomes

Buscando valores nulos na tabela

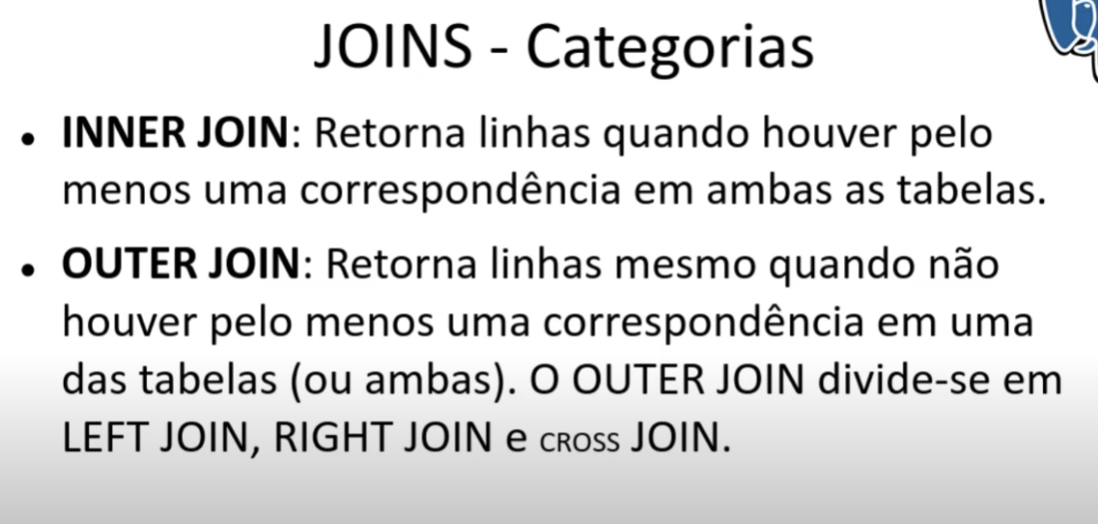
select customer.customer\_id,

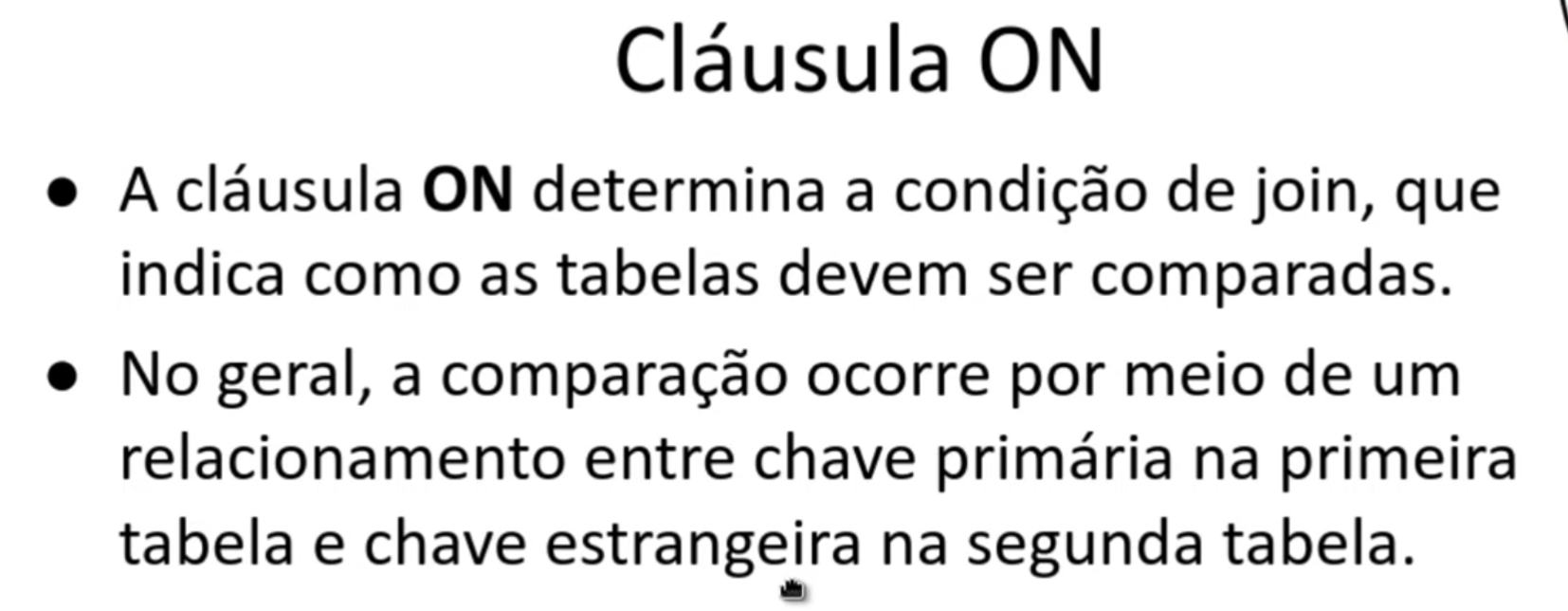
name as customer\_name

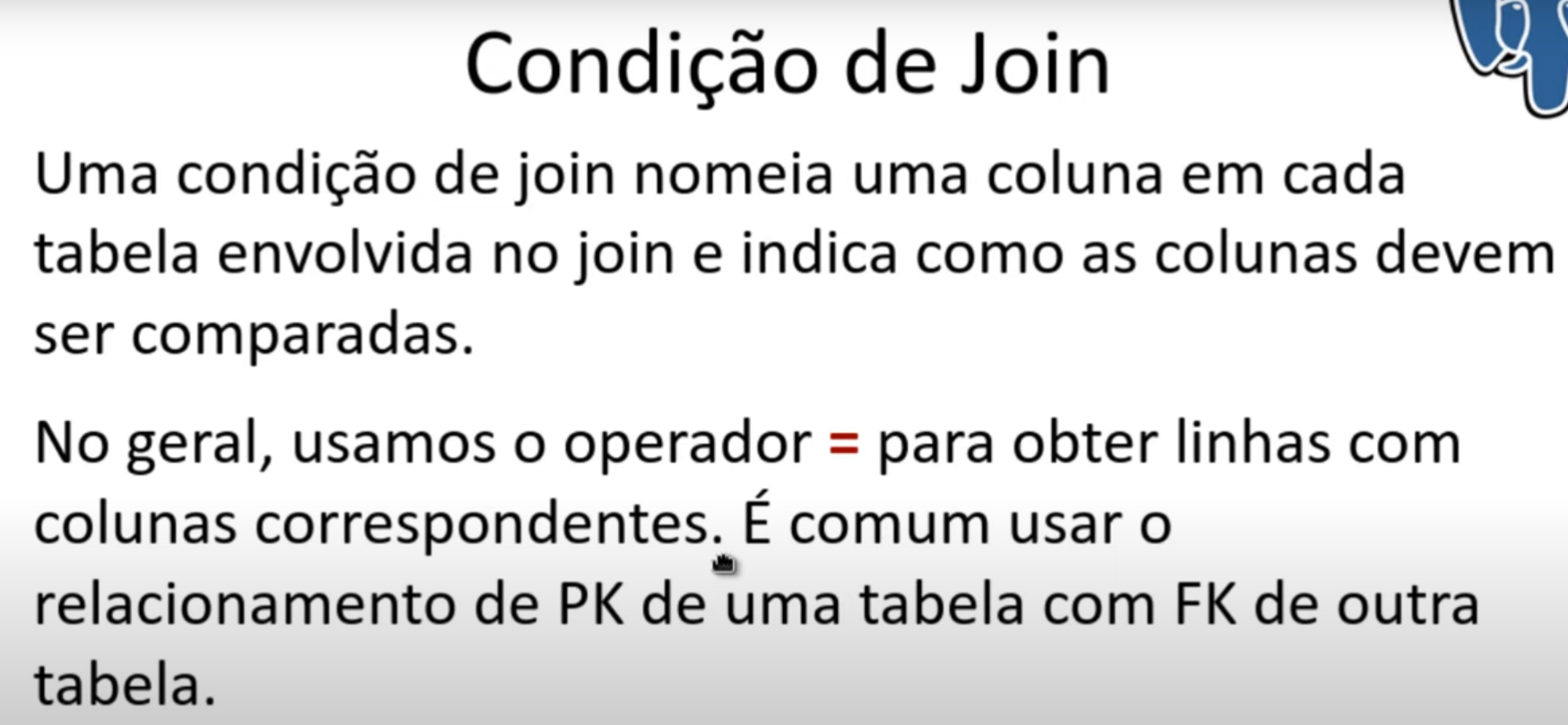
from customer left join customer order

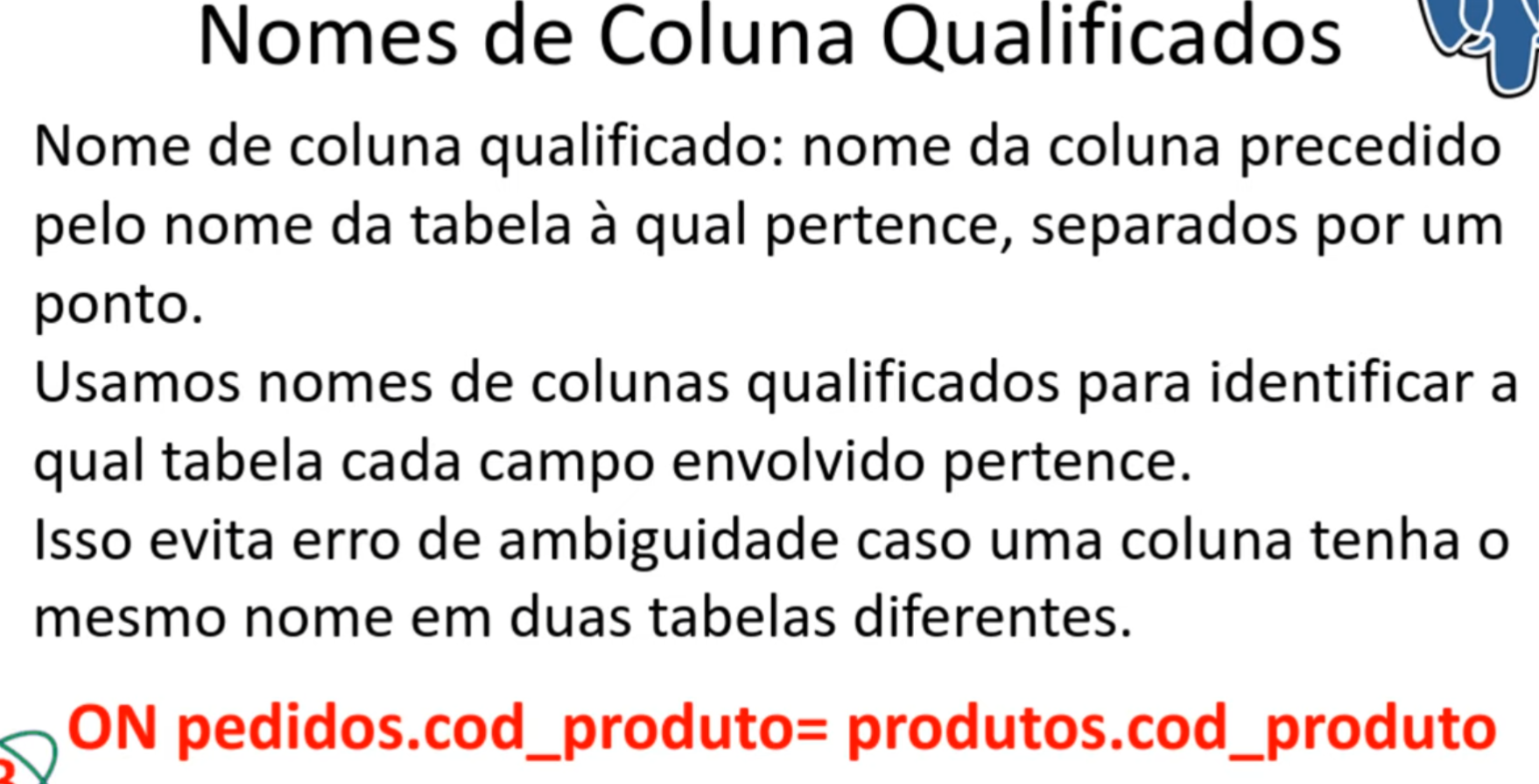
on customer.customer\_id = customer\_order.customer\_id

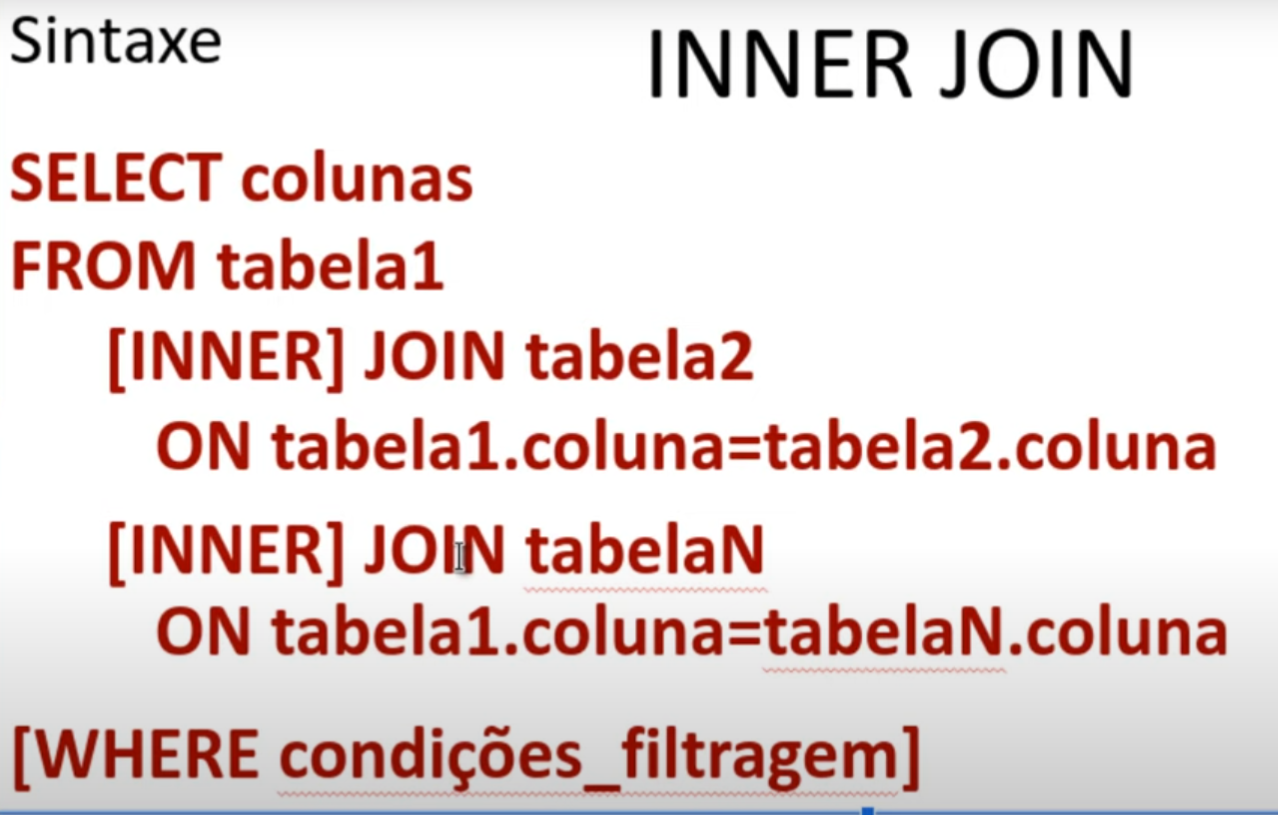
where order\_id is null



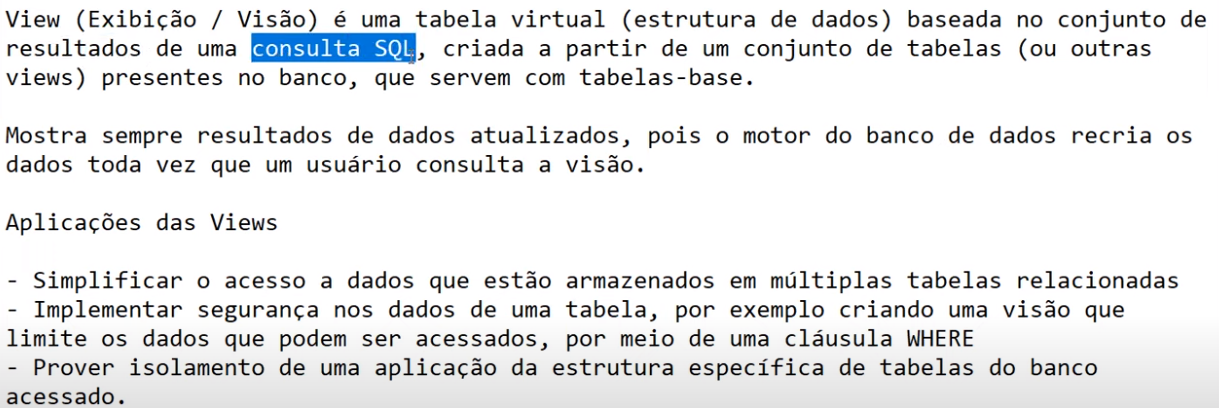








VIEWS



View: Tabela virtual (lógica) cujo conteúdo é baseado no retorno de uma consulta pré-definida, realizada em uma ou mais tabelas (ou outras view), mas que não contém os dados em si.

Simplificar as consultas e controlar o acesso aos dados de forma simplificada para os usuários com permissões.Permite criar uma visão mais lógica da modelagem do banco para os usuários.O conjunto de dados da view é grado no momento em que ela é executada.

Uma view é COMO SE FOSSE uma variável para consulta, voce cria essa view e ela aparece a consulta que sera feita

para criar uma view

CREATE OR REPLACE VIEW nome da viewr AS e em baixo o código da consulta

o OR REPLACE serve para se ja tiver uma view criada com o mesmo nome ele irá SUBSTITUIR essa view pelo que esta criando, exemplo na pratica para a criação dela

CREATE OR REPLACE VIEW vendas AS

SELECT CL.nome\_cliente AS Cliente, PR.nome\_produto AS Produto,

PE.qtde AS quantidade, PE.cod\_pedido AS Pedido,

PR.preco \* PE.qtde AS Fatura

FROM pedidos PE

INNER JOIN clientes CL

ON PE.cod\_cliente = CL.cod\_cliente

INNER JOIN produtos PR

ON PE.cod\_produto = PR.cod\_produto;

e para consultar pode fazer apenas um select

SELECT \* FROM vendas

tem como a gente alterar PROPRIEDADES da view

exemplo, pode renomear a view, apenas colocar

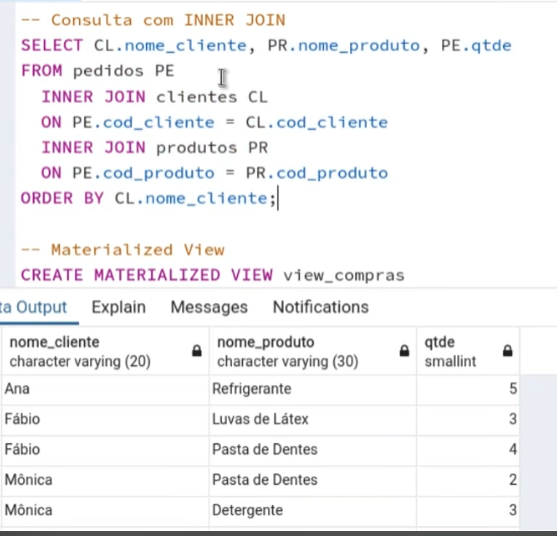
ALTER VIEW vendas RENAME TO faturas

e para excluir pode usar um DROP VIEW nome da view

Materialed View : é um objeto de banco de dados que armazena o resultado de uma consulta de forma persistente.É uma tabela auxiliar que permite maior performance no acesso aos dados. Uma materialized view (visão materializada) é uma cópia persistente e armazenada dos dados de uma consulta. Diferente de uma visão padrão (ou view), que executa a consulta toda vez que é acessada, a materialized view armazena os resultados da consulta no momento em que é criada. Isso pode melhorar significativamente o desempenho em consultas complexas ou que exigem agregações extensivas, porque os dados já estão pré-computados e armazenados.

no caso ela esta armazenada a consulta dos dados, ele ja cria em uma matéria

exemplo de uso



No final o with NO DATA serve para criar um materilized view sem os dados, e se retirar o NO ele cria com os dados

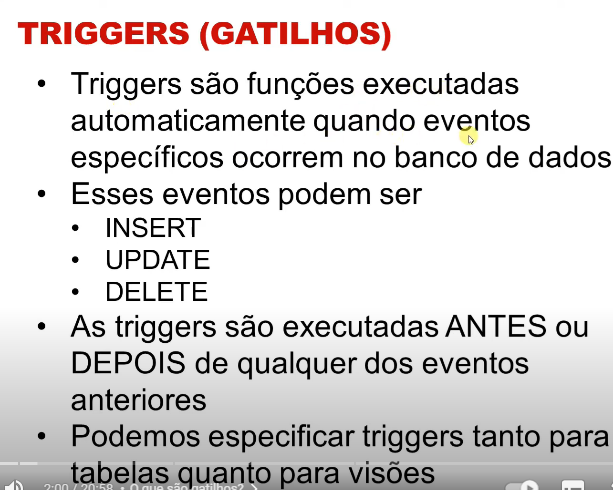
REFRESH MATERIALIZED VIEW nome\_da\_materialized\_view;

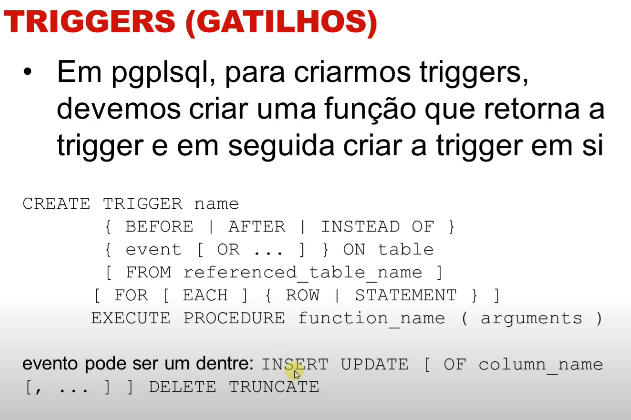
Como os dados em uma materialized view são estáticos e não são atualizados automaticamente quando os dados das tabelas subjacentes mudam, é necessário atualizar ou "refrescar" a materialized view para garantir que ela reflita os dados mais recentes.

pode tambem alterar um materialized view apenas colocando

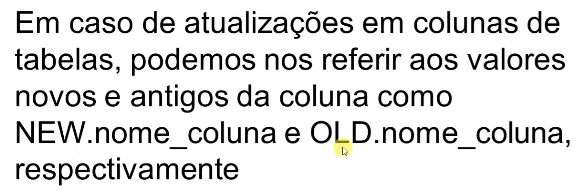
ALTER MATERIALIZED VIEW nome da view,

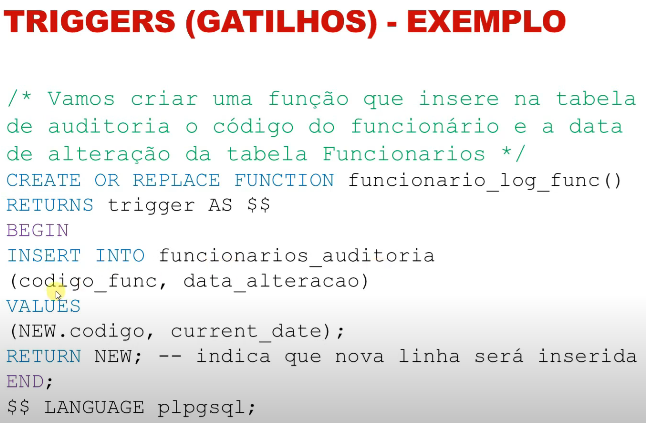
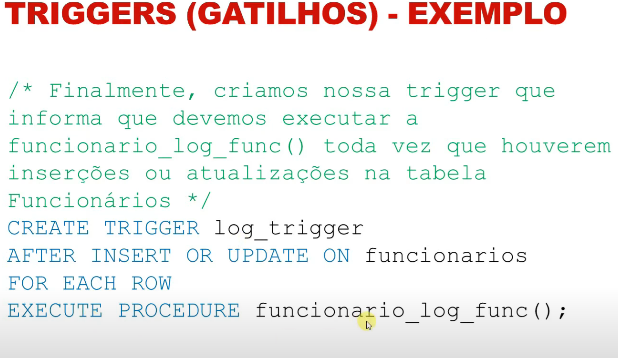
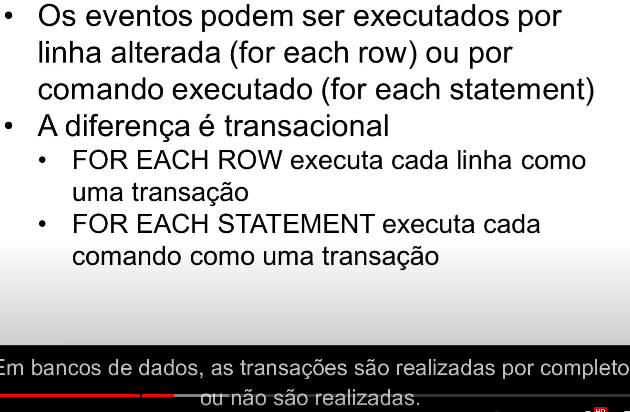
TRIGGER

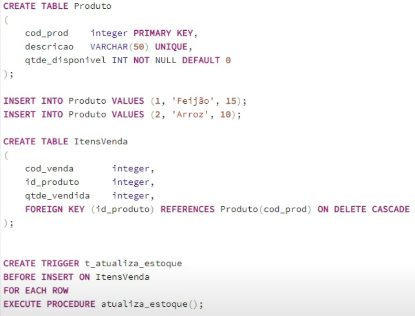




Pode ser usados em View







funções:

instr(): verifica se uma string de texto contém outra string de texto; se contiver, retorna o índice da posição em que ela foi encontrada.|Caso contrário, retorna 0

trim(): Remove espaços desnecessários nos dois lados de uma String ex

trim(‘ x ’) vai sair x

tem a variações, ltrim para tirar os lados em branco esquerdo e rtrim para tirar os lados em branco direito.

random () retorna um numero aleatório

os índices são bastante utilizados para as pesquisas com select mais rapidas mas atrasa o insert update e delete, o recomendado é criar índices se estiver tendo problemas de desempenho na tabela, e quando a tabela for MUITO grande

BEGIN TRANSACTION OU BEGIN

serve para fazer transações de informação depois que da este comando qualquer instrução de insert,update ou delete sera registrada para que possa ser cancelada se necessario, assim não perdendo o que foi feito basta chamar a instrução ROLLBACK que vai trazer aonde foi o ultimo begin ou commit

voce começa com START TRANSACTION;

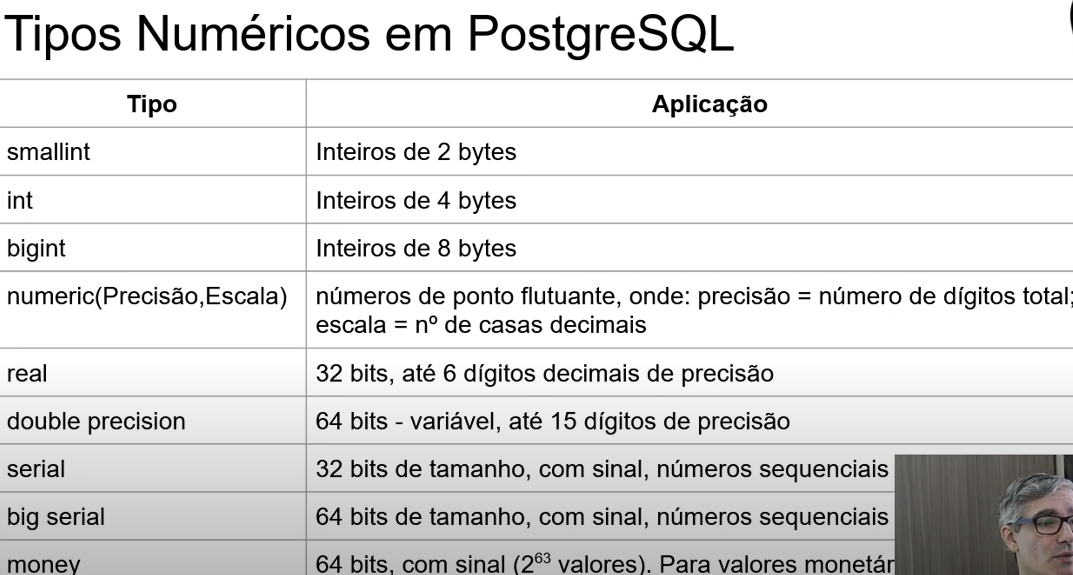
que inicia o programa

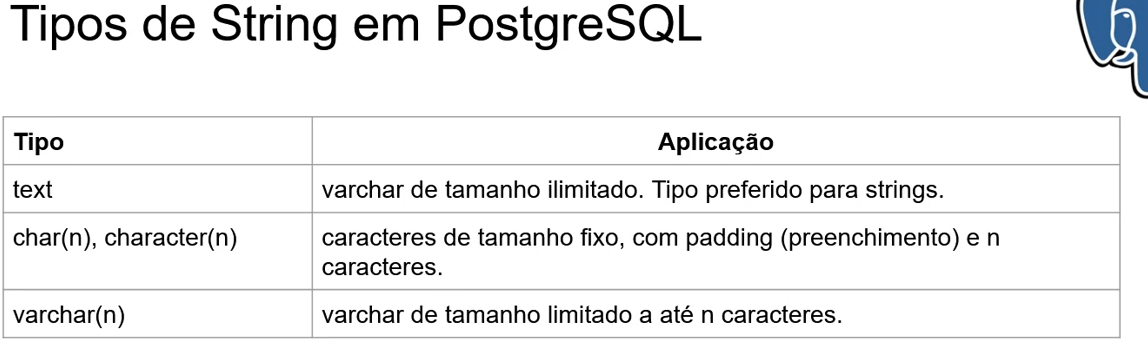
digite todos os comandos desejados e quando terminar escreva

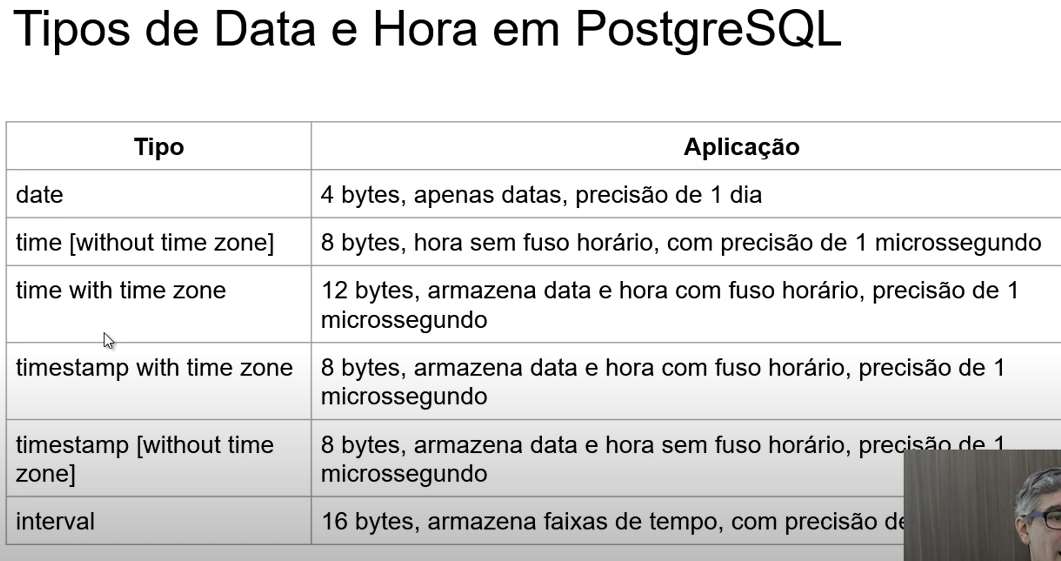
COMMIT;

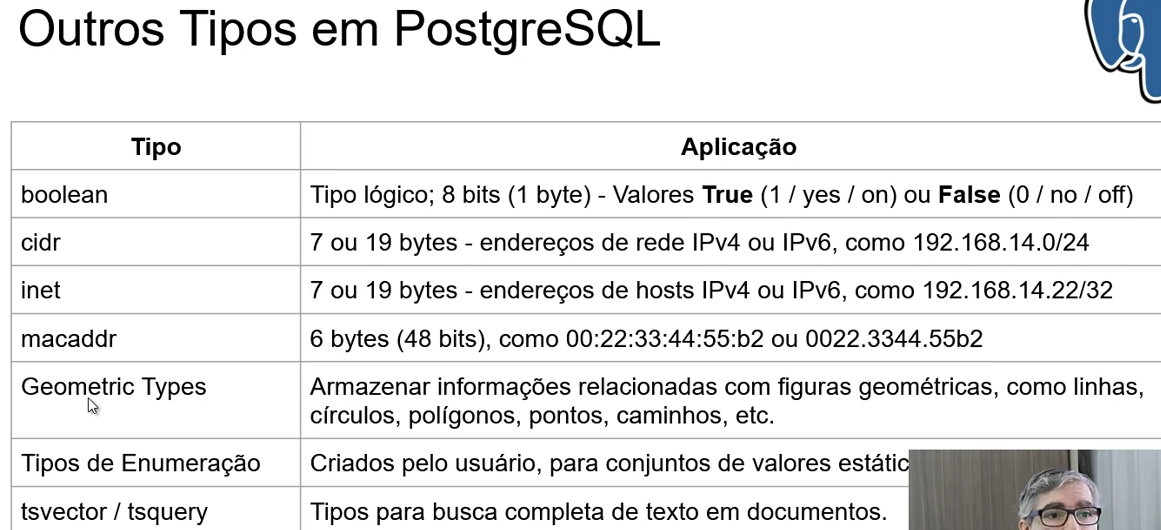
eles vão ser executados na ordem que foi escrito e no fim vai dar o commit

Postgresql









CHECK serve para garantir que vai digitar algum valor desejado, por exemplo

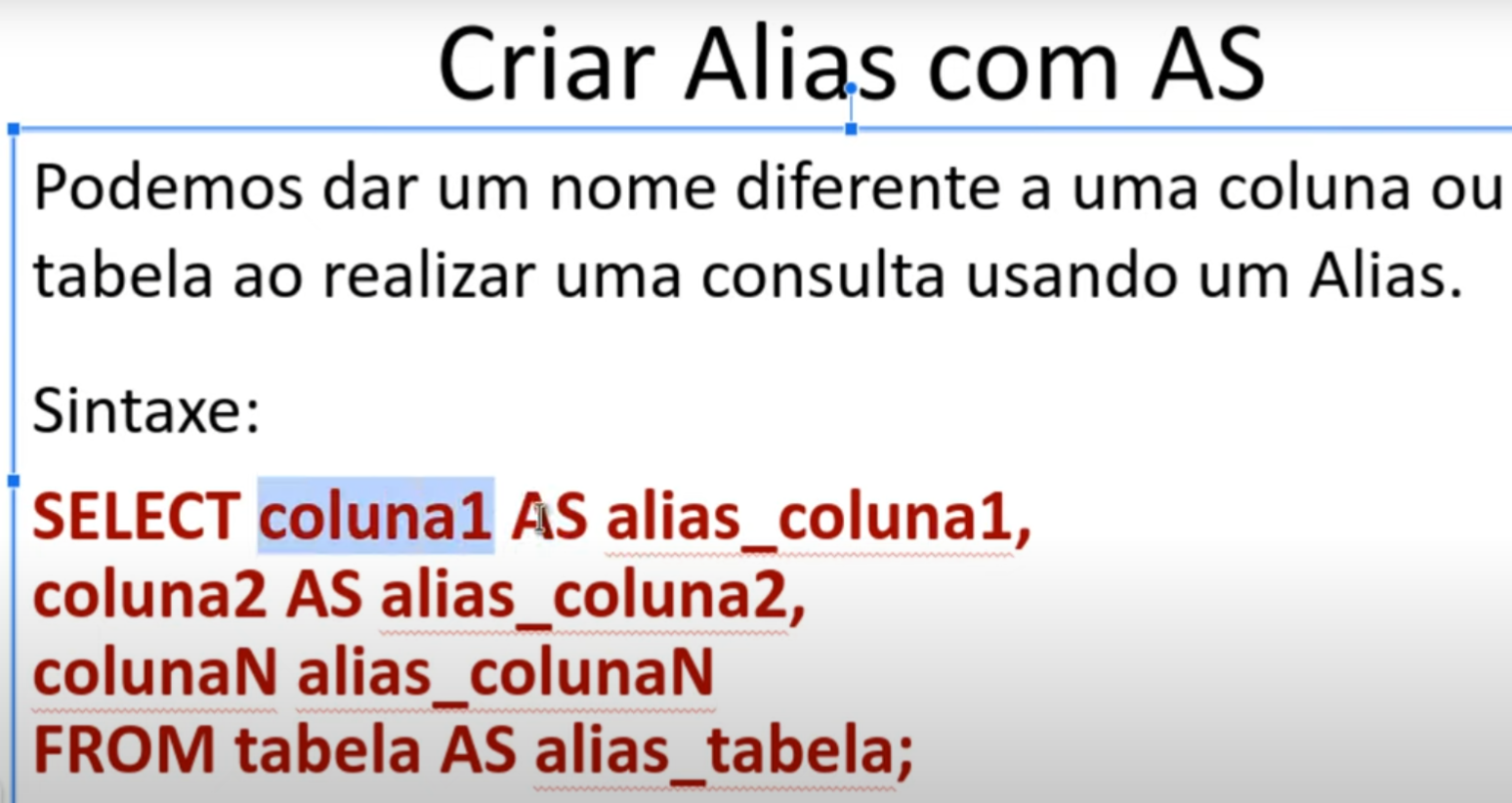
preco numeric check(preco>0) not null,

aqui vai informar que o preço precisa ser maior que 0

SERIAL, esse comando, faz a criação em sequencial de números inteiros sem precisar colocar. ex

cod\_pedido SERIAL PRIMARY KEY

AS



serve para APENAS A CONSULTA, o as será o nome que vai aparecer quando consultar

quando for pesquisar pode colocar com aspas para pesquisar

inherits ( xxx);

ela serve para pegar as colunas de uma tabela, e aplicar na criação de outra, ex

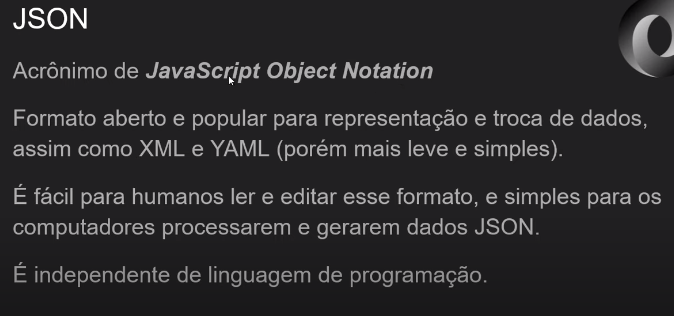
create table livros (

tipo varchar(20),

edição smallint

) inherits(publicação), vai pegar as colunas da tabela publicação e vai criar da mesma forma, sem preciar digitar tudo denovo

quando voce insere alguma informação nas tabelas filho, a tabela pai tambem puxa, mas se voce colocar na pai, não vai para as tabelas filho

json   


para declarar um arquivo json é apenas colocar o nome e o arquivo ex:

pedido json NOT NULL

no caso os registros precisam estar assim :

