[**https://www.vivaolinux.com.br/artigo/PostgreSQL-comandos-basicos**](https://www.vivaolinux.com.br/artigo/PostgreSQL-comandos-basicos)

sudo apt-get install postgresql

<https://www.postgresql.org/>

**Como criar o banco**

su – postgres

su postgres (para acessar o postgres dentro de um container docker)

OU

sudo -u postgres psql

CREATE DATABASE dbname;

psql -h localhost/OU/ip -p 5432 -U username dbname

CREATE USER rianwlp WITH ENCRYPTED PASSWORD 'xxx15159';

**psql -U <postgre username> -d <dbname>**

**Criando tabelas no PostgresSQL**

'CREATE TABLE users (idUsers SERIAL primary key,

name VARCHAR(255),

email VARCHAR(255) NOT NULL.

cpf INT (11) NOT NULL,

senha INT NOT NULL,

FOREIGN KEY (idNomeDaOutraTabela)references nomeDaOutraTabela(idNomeDaOutraTabela))');

**Como fazer as foreign Keys para todos os tipos de tabelas**

Importante sobre as tabelas

https://cursos.alura.com.br/course/introducao-postgresql-primeiros-passos/task/72585

**1x1**

CREATE TABLE tabela1 (idTabela1 SERIAL primary key,

Campos separados por virgula,

FOREIGN KEY (idNomeDaOutraTabela)references

nomeDaOutraTabela(idNomeDaOutraTabela)));

CREATE TABLE tabela2 (idTabela2 SERIAL primary key,

Campos separados por virgula,

FOREIGN KEY (idNomeDaOutraTabela)references

nomeDaOutraTabela(idNomeDaOutraTabela)));

**1x0**

CREATE TABLE tabela1 (idTabela1 SERIAL primary key,

Campos separados por virgula,

FOREIGN KEY (idNomeDaOutraTabela)references

nomeDaOutraTabela(idNomeDaOutraTabela)));

**1xn**

CREATE TABLE tabela1 (idTabela1 SERIAL primary key,

Campos separados por virgula,

FOREIGN KEY (idNomeDaOutraTabela)references

nomeDaOutraTabela(idNomeDaOutraTabela)));

CREATE TABLE tabela2 (idTabela2 SERIAL primary key)

**nxn**

CREATE TABLE criminoso (

idCriminoso SERIAL primary key,

nome VARCHAR(255),

cpf VARCHAR(11),

idade INT,

dataNascimento DATE,

FOREIGN KEY (idCriminoso) REFERENCES vitimaXcriminoso(idCriminoso),

);

CREAT TABLE vitimaXcriminoso (

idVitima SERIAL primary key,

idCriminoso SERIAL primary key

);

CREATE TABLE vitima (

idVitima SERIAL primary key,

nome VARCHAR(255),

cpf VARCHAR(11),

idade INT,

dataNascimento DATE,

FOREIGN KEY (idVitima) REFERENCES vitimaXcriminoso(idVitima),

);

**Tipos de condições que o PostgresSQL aceita**

!= OU <> Diferente > Maior < Menor = Igual >= Maior igual

<= Menor igual and or where between not

A cláusula WHERE pode ser combinada com os operadores AND, OR e NOT.

Os operadores AND e OR são usados ​​para filtrar registros com base em mais de uma condição:

O operador AND exibe um registro se todas as condições separadas por AND forem verdadeiras.

O operador OR exibe um registro se alguma das condições separadas por OR for TRUE.

O operador NOT exibe um registro se as condições NÃO forem verdadeiras.

**Como fazer inserts com o PostgresSQL**

Insert into nomeDaTabela (nomeDosCampos) VALUE (valores dos campos);

OU

Se você estiver adicionando valores para todas as colunas da tabela, não será necessário especificar os nomes das colunas na consulta SQL. No entanto, certifique-se de que a ordem dos valores esteja na mesma ordem das colunas da tabela. A sintaxe INSERT INTO seria a seguinte:

INSERT INTO *table\_name*  
VALUES (*value1*,*value2*,*value3*, ...);

NomeDaTabela campos que quero inserir valores dos campos

INSERT INTO nomeDaTabela (nome,email,senha,ativo) VALUES ('nome', 'email', senha','ativo')";

$sql = "INSERT INTO usuarios (nome,email,senha,ativo) VALUES ('nome', 'email', senha','ativo')";

$DB->prepare($sql)->execute($data);

Criar um tipo

Create type enumPagamento as enum (‘tiposDesejados’);

Alter table nomeTabela add column nomeDaColuna nomeDotipoCriado;

**Como fazer selects com o PostgresSQL**

Seleção simples:

select \* from nomeDaTabela

Para fazer um select em uma coluna com a condição where

select \* from nomeDaTabela where nomeDoCampo e codição

EXEMPLO:

select \* from comprar where valor > 1000;

Para fazer um select em mais de uma coluna ao mesmo tempo com a condição where

select \* from comprar where nomeDoPrimeiroCampo and outroCampo;

EXEMPLO:

select \* from comprar where valor > 1000 and data != ‘.../../….’ ;

Selects adiversos

select valor1,valor2,valor3 from nomeTabela where valor1 between 20 and 90 and not valor3 not = ola;

select valor,data,observacao from compras where data between ’01-01-2020 ’ and ’12-31-2020’ and not observacao = ‘gasto pequeno’

select to\_char(data,’YYYY’) as ano, sum(valor ) from compras group by ano;

ELE FAZ UM SELECT POR ANO, E SOMA O VALOR DAS COLUNAS E O ESCREVE NA TELA

IS NULL

Um campo com valor NULL é um campo sem valor.

Se um campo em uma tabela for opcional, é possível inserir um novo registro ou atualizar um registro sem adicionar um valor a este campo. Então, o campo será salvo com um valor NULL.

Sintaxe IS NULL

SELECT *column\_names*FROM *table\_name*  
WHERE *column\_name* IS NULL;

Sintaxe NÃO É NULL

SELECT *column\_names*FROM *table\_name*  
WHERE *column\_name* IS NOT NULL;

ORDER BY

A palavra-chave ORDER BY é usada para classificar o conjunto de resultados em ordem crescente ou decrescente.

A palavra-chave ORDER BY classifica os registros em ordem crescente por padrão. Para classificar os registros em ordem decrescente, use a palavra-chave DESC.

SELECT \* FROM diretor

ORDER BY nome;

SELECT \* FROM diretor

ORDER BY nome DESC;

MIN MAX

A função MIN () retorna o menor valor da coluna selecionada.

A função MAX () retorna o maior valor da coluna selecionada.

Sintaxe MIN ()

SELECT MIN(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

Sintaxe MAX ()

SELECT MAX(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

SQL COUNT (), AVG () e SUM ()

A função COUNT () retorna o número de linhas que correspondem a um critério especificado.

A função AVG () retorna o valor médio de uma coluna numérica.

A função SUM () retorna a soma total de uma coluna numérica.

Sintaxe COUNT ()

SELECT COUNT(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

Sintaxe AVG ()

SELECT AVG(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

Sintaxe SUM ()

SELECT SUM(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

SQL SELECT TOP

A cláusula SELECT TOP é usada para especificar o número de registros a serem retornados.

A cláusula SELECT TOP é útil em grandes tabelas com milhares de registros. Retornar um grande número de registros pode afetar o desempenho.

SELECT nome

FROM diretor

WHERE nome is not null

Coalesce

COALESCE (argument\_1, argument\_2, …);

Select

Coalesce (NULL, 1)

Substitui o null por 1, no caso o valor default

HAVING

SELECT

column1,

FROM

table\_name

GROUP BY

column1

HAVING

condition;

EXEMPLO:

SELECT

customer\_id,

COUNT (customer\_id)

FROM

payment

GROUP BY

customer\_id

HAVING

COUNT (customer\_id) > 40;

In this syntax, the group by clause returns rows grouped by the column1. The HAVING clause specifies a condition to filter the groups.

DATA\_TRUNC

date\_trunc('datepart', field)

SELECT DATE\_TRUNC('hour', TIMESTAMP '2017-03-17 02:09:30');

count()

O atributo count conta quantos tem em uma coluna, ou passado a condição para ele

SELECT

COUNT(DISTINCT column)

FROM

table\_name

WHERE condition;

Case When

CASE

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2 [WHEN ...]

[ELSE else\_result]

END

CASE

WHEN length> 0 AND length <= 50 THEN 'Short'

WHEN length > 50 AND length <= 120 THEN 'Medium'

WHEN length> 120 THEN 'Long'

END duration

GENERATE\_SERIES

generate\_series([start], [stop],[{optional}step/interval]);

EXEMPLO:

SELECT data\_inicial::DATE AS data\_inicial,

(data\_inicial + '1 month'::INTERVAL - '1 day'::INTERVAL )::DATE data\_final

FROM GENERATE\_SERIES((SELECT data\_inicial FROM datas), (SELECT data\_final FROM datas), '1 month'::INTERVAL) data\_inicial

Replace

REPLACE(source, old\_text, new\_text );

Exemplo:

SELECT

REPLACE ('ABC AA', 'A', 'Z');

Exists

EXISTS (subquery)

SELECT column1

FROM table\_1

WHERE

EXISTS( SELECT 1

FROM

table\_2

WHERE

column\_2 = table\_1.column\_1);

Para fazer um select em duas colunas ao mesmo tempo sem nenhuma condição

select nomeDaColuna, nomeDaColuna from nomeDaTabela;

===========================================

Para fazer um select em uma só coluna

select nomeDaColuna from nomeDaTabela

Pode ser feito operações logicas no banco por exemplo

EXEMPLO:

select valor\*0.2 as Imposto from comprar

Neste caso ele irá criar uma coluna com o nome de Imposto na tabela compras

===========================================

Para fazer um select em uma só coluna alterando o nome dela

select nomeCampo AS “Nome novo’

EXEMPLO:

SELECT nome as “Nome do aluno”,

idade,

matriculado AS quandoSeMatriculou

FROM aluno;

Para fazer um select quando você não sabe o dado inteiro

select \* from nomeDaTabela where nomeDoCampo like ‘combustivel%’;

OU

select \* from nomeDaTabela where nomeDoCampo like ‘%combustivel’;

Para fazer um select em mais de uma tabela

<https://cursos.alura.com.br/course/introducao-postgresql-primeiros-passos/task/72583>

select \* nomeTabela, nomeOutraTabela where condição;

EXEMPLO:

select \* compras, loja where lojasId = lojas.id;

EXEMPLO COM JOIN

SELECT aluno.nome, curso.nome FROM aluno

OU

SELECT \* from aluno

===========================================

INNER JOIN

SELECT \* FROM parcela\_um

INNER JOIN parcela\_dois ON parcela\_um.curso = parcela\_dois.curso

===========================================

LEFT JOIN

LEFT JOIN alunoCurso ON alunoCurso.alunoId = alunoId

LEFT JOIN curso ON cursoId = alunoCurso.cursoId

TEM DADO NA TABELA DA ESQUERDA MAS NÃO TEM DADO NA TABELA DA DIREITA

===========================================

RIGHT JOIN

RIGHT JOIN alunoCurso ON alunoCurso.alunoId = alunoId

RIGHT JOIN curso ON cursoId = alunoCurso.cursoId

TEM DADO NA DIREITA MAS NÃO TEM NA ESQUERDA

===========================================

FULL JOIN

FULL JOIN alunoCurso ON alunoCurso.alunoId = alunoId

FULL JOIN curso ON cursoId = alunoCurso.cursoId

NÃO IMPORTA SE TIVER DADO OU NÃO ELE VAI TRAZER OS DOIS NULL OU NÃO NULL

===========================================

CROSS JOIN

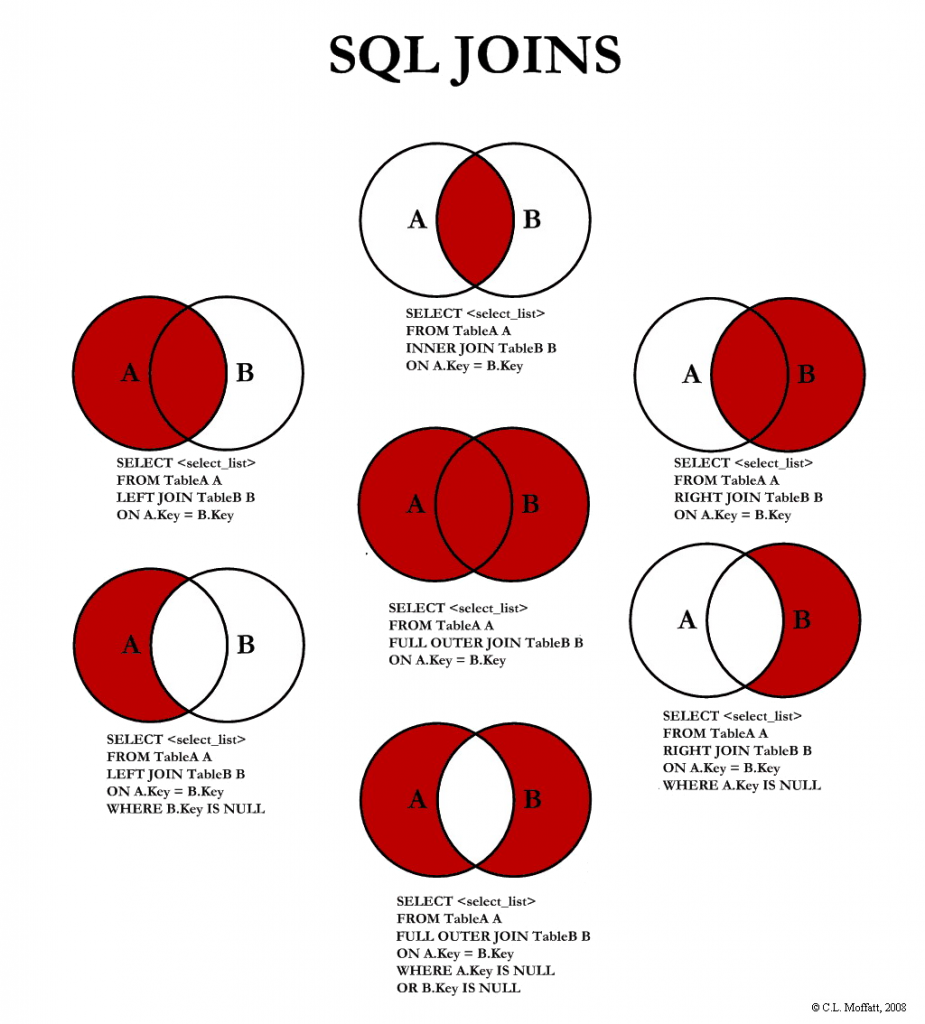
SELECT \* from aluno.nome, curso.nome

FROM aluno

CROSS JOIN curso

PEGA OS ALUNOS E COLOCA TODOS OS CURSOS PARA CADA ALUNO

===========================================



UNION

SELECT column1 [, column2 ]

FROM table1 [, table2 ]

[WHERE condition]

UNION

SELECT column1 [, column2 ]

FROM table1 [, table2 ]

[WHERE condition]

===========================================

LIMIT

Restringe a quantidade de registros trazidos do banco

EXEMPLO:

SELECT\* FROM funcionarios LIMIT 5;

OU AINDA

SELECT\* FROM funcionarios ORDER BY nome LIMIT 5;

OU AINDA PEGAR DE UM ID PARA FRENTE

EXEMPLO:

SELECT\* FROM funcionarios ORDER BY nome LIMIT 5 OFFSET 3;

Ira pegar do três para frente

===========================================

DISTINCT

SELECT DISTINCT nome FROM funcionarios ORDER BY nome;

Irá ordenar por nome, se tiver um nome repetido ele será agrupado assim não sendo mostrado e mostrando apenas um

===========================================

WITH

WITH name\_for\_summary\_data AS (

SELECT Statement)

SELECT columns

FROM name\_for\_summary\_data

WHERE conditions <=> (

SELECT column

FROM name\_for\_summary\_data)

[ORDER BY columns]

Exemplo:

WITH parametro as (

SELECT

CASE WHEN length('${kanban\_tecnico}') > 0 THEN '$ {kanban\_tecnico}' ELSE '0' END as tecnico,

CASE WHEN length('${kanban\_cliente}') > 0 THEN '$ {kanban\_cliente}' ELSE '0' END as cliente,

CASE WHEN length('${kanban\_tipo}') > 0 THEN '$ {kanban\_tipo}' ELSE '0' END as tipo,

CASE WHEN length('${kanban\_logado\_como}') > 0 THEN '$ {kanban\_logado\_como}' ELSE '0' END as logado

)

===========================================

**Comandos do Postgres no terminal**

\d nome da tabela Mostra a tabela no terminal

\i ~ / caminho do documento Importa dados para o banco

\d Lista tabelas do banco

**Comando Update**

update nomeDaTabela set nomeDaColuna = valor where id = idDaLinha;

EXEMPLO:

Neste exemplo ele atualiza todos os registros que tiverem valor menor que 20

update compras set observacao = ‘gasto pequeno’ where valor < 20;

UPDATE *table\_name*  
SET *column1*=*value1*,*column2*=*value2*, ...  
WHERE *condition*;

Update elementos quando tem foreign key

CREATE TABLE pessoas (

id INTEGER PRIMARY KEY,

nome VARCHAR(255) NOT NULL

);

CREATE TABLE telefones (

id INTEGER PRIMARY KEY,

pessoa\_id INTEGER,

numero VARCHAR(15) NOT NULL,

FOREIGN KEY (pessoa\_id) REFERENCES pessoas (id)

ON DELETE RESTRICT

ON UPDATE CASCADE

);

FOREIGN KEY (idVitima) REFERENCES vitimaXcriminoso(idVitima) ON DELETE CASCADE

OU

FOREIGN KEY (idVitima) REFERENCES vitimaXcriminoso(idVitima) ON UPDATE CASCADE

**Deletar elementos da tabela**

delete from nomeTabela where valor = ‘Rian’

EXEMPLO:

delete from comprar where data between ’01-02-2020’ and ’03-09-2021’;

DELETE FROM *table\_name*WHERE *condition*;

**Nota:** Tenha cuidado ao excluir registros em uma tabela! Observe a cláusula WHERE na instrução DELETE. A cláusula WHERE especifica quais registros devem ser excluídos. Se você omitir a cláusula WHERE, todos os registros na tabela serão excluídos!

Deletar elementos quando tem foreign key

FOREIGN KEY (idVitima) REFERENCES vitimaXcriminoso(idVitima) ON DELETE CASCADE

Agora quando ele tentar deletar o aluno e o aluno tiver vinculo em outra tabela ele irá excluir o vinculo e deletar o aluno

**Alter table**

Renomear uma tabela

Alter table comprar rename nomeAntigo to nomeNovo;

Para criar uma foreign key

Alter table nomeDaTabela add foreign key (idDaTabelaReferenciada) references (idDaTabelaReferenciada);

**FOREIGN KEY (coluna) REFERENCES tabela-pai (coluna);**

Para alterar uma coluna

Alter table nomeTabela alter COLUMN nomeColuna set NOT NULL;

Para criar uma coluna

Alter table nomeTabela add column nomeDoCampo tipo do campo

EXEMPLO:

Alter table users add column nome VARCHAR (255)

**Tipos de dados mais usados no postgresSQL**

* SERIAL
* VARCHAR
* CHAR
* TEXT
* INTEGER ou int
* NUMERIC
* BOOLEAN
* DATE
* TIME
* TIMESTAMP

**Instalando o PostgresSQL no Linux**

https://cursos.alura.com.br/course/introducao-a-banco-de-dados-e-sql-com-postgre/task/18717

sudo apt-get update

sudo apt-get -y install postgresql postgresql-contrib

$ sudo -u postgres psql

postgres=# \password postgres