Introdução

Introdução - comandos Postgresql

- “.sql” é que todos servem a mesma padronização.Se você jogar o código de um SGBD para o outro ele vai funcionar
- O “USE” não funciona no Postgresql
- O postgree não aceita alias como string.Ou seja, tire as aspas

TIPOS DE DADOS

- **MONEY** = Float
- **SERIAL** = int AUTO_INCREMEN
- **CHARACTER VARYING**= VARCHAR (comprimento variável)
- **CHAR** = Tamanhos pequenos/fixos
- **DATE** = data

CRIANDO TABELAS

```
#Criação de banco de dados
CREATE DATABASE secao03;

#Criando tabela
CREATE TABLE tipos_produto(
    id SERIAL PRIMARY KEY,
    descricao CHARACTER VARYING (50) NOT NULL
);

CREATE TABLE produtos(
    id SERIAL PRIMARY KEY,
    descricao CHARACTER VARYING (50) NOT NULL,
    preco MONEY NOT NULL,
    id_tipo_produto INT REFERENCES tipos_produto id NOT NULL;
);
```

```
CREATE TABLE pacientes(
    id SERIAL PRIMARY KEY,
    nome CHARACTER VARYING (50) NOT NULL,
    endereco CHARACTER VARYING (50) NOT NULL,
    bairro CHARACTER VARYING (50) NOT NULL,
    cidade VARCHAR (40) NOT NULL;
    estado CHAR (2) NOT NULL,
    cep VARCHAR (5) NOT NULL,
    data_nasc DATE NOT NULL

);

CREATE TABLE professores (
    id SERIAL PRIMARY KEY,
    telefone INT NOT NULL,
    nome VARCHAR (50) NOT NULL,


);

CREATE TABLE turma (
    id SERIAL PRIMARY KEY,
    capacidade INT NOT NULL,
    id_professor INT REFERENCES professores(id) NOT NULL
);
```

OPERADORES

```
= (igual a): Compara se um valor é igual a outro.

Exemplo: SELECT * FROM tabela WHERE coluna = 10;



< (menor que): Compara se um valor é menor que outro.

Exemplo: SELECT * FROM tabela WHERE coluna < 10;
```

```
<= (menor ou igual a): Compara se um valor é menor ou igual a

Exemplo: SELECT * FROM tabela WHERE coluna <= 10;


> (maior que): Compara se um valor é maior que outro.

Exemplo: SELECT * FROM tabela WHERE coluna > 10;


>= (maior ou igual a): Compara se um valor é maior ou igual a

Exemplo: SELECT * FROM tabela WHERE coluna >= 10;


<> (diferente de): Compara se um valor é diferente de outro.

Exemplo: SELECT * FROM tabela WHERE coluna <> 10;


AND (E):

Exemplo: SELECT * FROM animais WHERE altura > 25 AND especies


OR (ou):

Exemplo: SELECT * FROM animais WHERE altura > 25 OR especies
```

RENOMEAR A TABELA ORIGINAL

```
ALTER TABLE clientes RENAME TO clientes_backup;
```

ADICIONANDO UMA CHAVE PRIMÁRIA A UMA TABELA | CHAVE SUBSTITUTA | CHAVE ESTRANGEIRA

- Adicionar uma **chave primária** a uma tabela existente no PostgreSQL

```
ALTER TABLE table_name ADD CONSTRAINT some_name PRIMARY KEY (

ALTER TABLE table_name ADD COLUMN id SERIAL PRIMARY KEY; #esp

ALTER TABLE table_name #Especifica a tabela à qual você desej
ADD CONSTRAINT constraint_name #Define o nome da restrição de
FOREIGN KEY (fk_column_name) #Especifica a tabela que servirá
REFERENCES referenced_table_name (referenced_column_name);  #
```

INSERINDO DADOS NA TABELA | FROM | ALIAS

- Mesma coisa do mysql

```
INSERT INTO tipos_produto (descricao) VALUES ("Computadores")
INSERT INTO tipos_produto (descricao ) VALUES ("Acessórios");
INSERT INTO tipos_produtos (descricao) VALUES ("Diversos");
```

```
INSERT INTO produtos (descricao, preco, id_tipos_produto) VAL
INSERT INTO produtos (descricao, preco, id_tipos_produto) VAL
INSERT INTO produtos (descricao, preco, id_tipos_produto) VAL
```

```
#Select
SELECT * FROM produtos;
SELECT descricao, preco FROM produtos;
```

```
#Alias
SELECT p.codigoo AS cod, p.descricao AS descr, p.preco AS pre
```

INSERINDO COLUNAS

```
ALTER TABLE table_name ADD COLUMN column_name column_type;
```

UPDATE e DELETE

```
UPDATE tipos_produto SET descricao = "Nobreak" WHERE codigo =
```

```
DELETE FROM tipos_produto WHERE codigo = 2; #Deletando a linh
```

ALTER e DROP

```
ALTER TABLE table_name ADD peso DECIMAL (0,2); //adiciona um
```

```
ALTER TABLE table_name RENAME COLUMN idade TO Idade_anos //Re
```

```
ALTER TABLE table_name DROP COLUMN column_name //Alterar tabe

ALTER TABLE table_name ALTER COLUMN column_name TYPE varchar(

ALTER TABLE table_name ALTER COLUMN column_name SET NOT NULL

ALTER TABLE table_name ALTER COLUMN column_name DROP NOT NULL
```

```
DROP TABLE tipos_produto; //deleta a tabela

DROP DATABASE secao04; //deleta o BD
```

TRANSAÇÃO E ROLLBACK

- **Rollback:** remove todas as alterações feitas desde a última operação de confirmação ou rollback.
  - Não adianta executar o rollback tendo o commit (comente-o #)

```
#Transação
BEGIN TRANSACTION

INSERT INTO tipos_produto (descricao) VALUES ("Equipamentos")
```

```
COMMIT;



#Rollback
BEGIN TRANSACTION

INSERT INTO tipos_produto (descricao) VALUES ("Equipamentos")


ROLLBACK;
```

FILTRAGEM COM O WHERE e CONSULTA COM MÚLTIPLAS TABELAS | DESCOBRINDO O MAIOR CARACTERE

```
#Forma 1 - Simples
SELECT * FROM tipos_produto WHERE codigo = 2;
SELECT * FROM tipos_produto WHERE preco <= 500;

#Multiplas Tabelas
SELECT p.codigo AS "Código", p.descricao AS "Descrição", p.pr
    FROM produto AS p , tipos_produto AS tp;
    WHERE p.codigo_tipo = tp.codigo;
```

```
SELECT MAX(CHAR_LENGTH(nome)) AS maior_caractere FROM produto
```

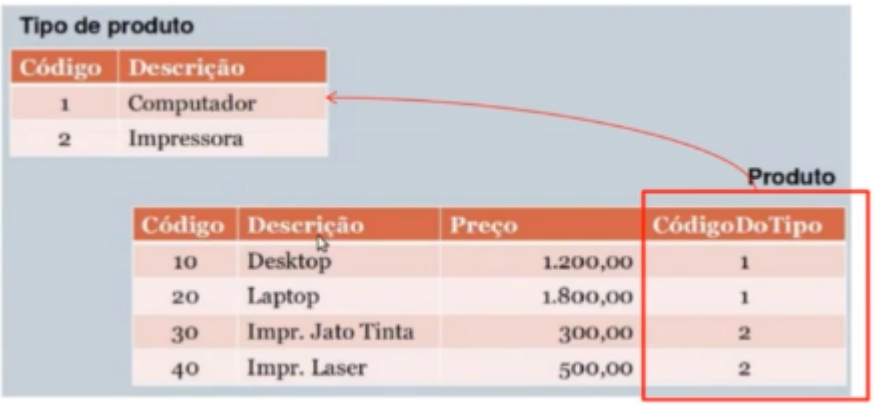

Figura 1: Veja que a tabela “Produto” está relacionada com a tabela “Tipo de produto”.Imagine que você queira trazer os seguintes dados.. “codigo do produto, descrição do produto, preço do procuto e descrição do tipo de produto”

JUNÇÃO DE TABELAS

```
#Junção de produto cartesiano - mesma coisa do 'multiplas tab
SELECT c.id, c.nome, c.data_nascimento, c.telefone, p.cargo
    FROM clientes AS c, profissoes AS p
    WHERE c.id_profissao = p.id;

#Inner Join
SELECT c.id, c.nome, c.data_nascimento, c.telefone, p.cargo
    FROM clientes AS c INNER JOIN profissoes AS p
    ON c.id_profissao = p.id;


#Left Outer Join
SELECT * FROM clientes
LEFT OUTER JOIN profissoes
ON clientes.id_profissao = profissoes.id;

#Right Outer Join
SELECT * FROM clientes
RIGHT OUTER JOIN profissoes
ON clientes.id_profissao = profissoes.id;

#Full Alter Join
```

```
SELECT * FROM clientes
FULL OUTER JOIN profissoes
ON clientes.id_profissao = profissoes.id;

#Full Alter Join 2
SELECT * FROM clientes
LEFT OUTER JOIN profissoes
ON clientes.id_profissao = profissoes.id;
UNION
SELECT * FROM clientes
RIGHT OUTER JOIN profissoes
ON clientes.id_profissao = profissoes.id;
```

AGREGAÇÃO e AGRUPAMENTO

1. **Média (`AVG`)**
   - Calcula a média dos valores de uma coluna.

2. **Arredondar valores (`ROUND`)**
   - Arredonda números para um número específico de casas decimais.

3. **Contagem (`COUNT`)**
   - Conta o número de registros que atendem a uma condição específica.

4. **Agrupamento (`GROUP BY`)**
   - Agrupa linhas que têm valores em comum e aplica funções de agregação, como somar ou contar.

5. **Filtrar grupos com `HAVING`**
   - Filtra grupos após a agregação, diferente do `WHERE`, que filtra antes do agrupamento. Ou seja, o HAVING vem junto com o GROUP BY, e ambos

sempre vem antes do WHERE

6. **Ordenação (`ORDER BY`)**
   - Ordena os resultados em ordem crescente (`ASC`) ou decrescente (`DESC`).

```
#Formatação
SELECT AVG(preco_custo) FROM produtos;
SELECT TO_CHAR(AVG (preco_venda), '999999999') as media FROM
SELECT ROUND(AVG (preco_venda)::numeric, 2) as media FROM pro
SELECT ROUND(AVG (preco_custo)::numeric), 2) as media FROM pr

#Média

-- Exemplo 1 Calcular a média salarial dos funcionários de ca
SELECT e.pais, AVG(p.salario) AS media_salarial
FROM funcionarios AS f
JOIN pagamentos AS p ON p.id_funcionario = f.id
JOIN escritorio AS e ON f.id_escritorio = e.id
GROUP BY e.pais;

-- Exemplo 2
SELECT f.nome, f.sobrenome, e.pais, p.salario
FROM pagamentos AS p, funcionarios AS f, escritorio AS e
WHERE f.id_escritorio = e.id
  AND f.id = p.id_funcionario
  AND p.salario < (SELECT AVG(salario) FROM pagamentos);



#Arredondar

-- Exemplo 1
SELECT ROUND(AVG(preco_venda), 4) FROM produtos;

-- Encontre a média dos salários dos funcionários de cada esc
SELECT e.pais, ROUND(AVG(p.salario), 2) AS media_arredondada
```

```
FROM pagamentos AS p
JOIN funcionarios AS f ON p.id_funcionario = f.id
JOIN escritorio AS e ON f.id_escritorio = e.id
GROUP BY e.pais;



#Contagem

-- Exemplo 1
SELECT COUNT(preco_venda) AS Quantidade FROM produtos WHERE i

-- Exemplo 2: Conte o número de funcionários em cada escritór

SELECT e.pais, COUNT(f.id) AS numero_funcionarios
FROM funcionarios AS f
JOIN escritorio AS e ON f.id_escritorio = e.id
GROUP BY e.pais;



#Group By
SELECT t.nome AS Tipo, f.nome AS Fabricante, SUM(p.quantidade
    FROM tipos AS t, fabricantes AS f, produtos AS p
        WHERE t.id = p.id_tipo AND f.id = p.id_fabricante
        GROUP BY t.nome, f.nome
        HAVING SUM(p.quantidade) > 200;




#Having

-- Exemplo 1
SELECT id_categoria, MAX (preco_venda) FROM produtos GROUP BY

-- Exemplo 2
SELECT e.pais, COUNT(f.id) AS numero_funcionarios
FROM funcionarios AS f
```

```
JOIN escritorio AS e ON f.id_escritorio = e.id
GROUP BY e.pais
HAVING COUNT(f.id) > 2;




#ORDER BY DESC e ASC
SELECT id, nome, id_tipo, id_fabricante, quantidade FROM prod
```

FUNÇÕES DE DATA , HORA E FORMATAÇÃO

```
#Data e hora atual
SELECT NOW() AS Data_hora;

#Data atual
SELECT CURRENT_DATE AS data_atual;

#Hora atual
SELECT CURRENT_TIME AS hora atual;

#Calcular data futura
SELECT CURRENT_DATE + INTERVAL '3 DAY' AS data_vencimento;

#Calcular data passada
SELECT CURRENT_DATE - INTERVAL '3 DAY' AS data_passada;

#Em anos
SELECT DATE_PART('year', '20-01-01':: date) - DATE_PART('year

#Em meses
SELECT (DATE_PART('year', '2019-01-01'::date) - DATE_PART('ye
    (DATE_PART('month', '2019-01-01'::date) - DATE_PART('mont

#Em semanas
```

```
SELECT TRUNC(DATE_PART('day', '2019-01-01'::timestamp - '2011

#Em dias
SELECT DATE_PART('day', '2019-01-01'::timestamp - '2011-10-02

#Em horas
SELECT DATE_PART('day', '2019-01-01 11:55'::timestamp - '2019
    DATE_PART('hour', '2019-01-01 11:55'::timestamp - '2019-0

#Em minutos
SELECT DATE_PART('day', '2019-01-01 11:55'::timestamp - '2019
    DATE_PART('hour', '2019-01-01 11:55'::timestamp - '2019-0
    DATE_PART('minute', '2019-01-01 11:55'::timestamp - '2019

#Formatando data no postgresql
SELECT TO_CHAR(CURRENT_DATE, 'dd/m/yyyy') AS Data_Atual;

#Com inicial minuscula - funciona a mesma coisa para mes
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'Day') AS Dia_da_Semana);

#Com short maiuscula
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'Dy') AS Dia_da_semana;

#Com inicial minuscula
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'dy') AS Dia_da_semana;

#Tudo maiuscula
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'DAY') AS Dia_da_semana;
```

```
#Transformar em pt-br
SHOW lc_time;
SET lc_time='pt_BR.UTF8';

#Exemplo
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'TMDay') AS Dia da semana;
```

```
#Convertendo de segundos para hora
SELECT TO_CHAR((2000 || 'seconds')::interval, 'HH24:MM:SS') A
```

SUBCONSULTA E SEM SUBCONSULTA

- Para resolver a subconsulta é o seguinte:

1. Consulta Principal: Você começa criando a consulta principal que traz os **dados principais que você precisa** (p.post_id, p.content, COUNT(l.post_id) etc. , **as tabelas desses dados, bem como a relação**

```
SELECT p.post_id, p.content, COUNT(l.post_id) AS total_likes
FROM posts AS p
JOIN likes AS l ON p.post_id = l.post_id
WHERE p.user_id = (SELECT user_id FROM users WHERE username =
GROUP BY p.post_id, p.content
```

2. **Subconsulta**: serve para obter o valor com o qual você quer comparar na consulta principal. Aqui, você usa `ALL` para comparar com a maior contagem de curtidas

```
HAVING COUNT(l.post_id) >= ALL (
SELECT COUNT(l2.post_id)
FROM posts AS p2
JOIN likes AS l2 ON p2.post_id = l2.post_id
WHERE p2.user_id = (SELECT user_id FROM users WHERE username =
GROUP BY p2.post_id
);
```

```
-- create
CREATE TABLE escritorio(
  id SERIAL PRIMARY KEY,
  pais VARCHAR(30) NOT NULL

);
```

```
CREATE TABLE funcionarios(
  id SERIAL PRIMARY KEY,
  nome VARCHAR(20) NOT NULL,
  sobrenome VARCHAR(20) NOT NULL,
  id_escritorio INT REFERENCES escritorio(id) NOT NULL

);

CREATE TABLE pagamentos(
  id SERIAL PRIMARY KEY,
  id_funcionario INT REFERENCES funcionarios(id) NOT NULL,
  salario DECIMAL(8,2) NOT NULL,  -- Aumentando a precisão do
  data DATE NOT NULL
);

-- Inserindo dados nas tabelas
INSERT INTO escritorio(pais) VALUES ('Brasil');
INSERT INTO escritorio(pais) VALUES ('EUA');
INSERT INTO escritorio(pais) VALUES ('Alemanha');
INSERT INTO escritorio(pais) VALUES ('França');

INSERT INTO funcionarios (nome, sobrenome, id_escritorio) VAL
INSERT INTO funcionarios (nome, sobrenome, id_escritorio) VAL
INSERT INTO funcionarios (nome, sobrenome, id_escritorio) VAL
INSERT INTO funcionarios (nome, sobrenome, id_escritorio) VAL

INSERT INTO pagamentos(id_funcionario, salario, data) VALUES
INSERT INTO pagamentos(id_funcionario, salario, data) VALUES
INSERT INTO pagamentos(id_funcionario, salario, data) VALUES
INSERT INTO pagamentos(id_funcionario, salario, data) VALUES


-- Exemplo 1: SubConsulta SQL para obter nomes e sobrenomes d
SELECT nome, sobrenome
FROM funcionarios
WHERE id_escritorio IN (
    SELECT id
```

```
    FROM escritorio
    WHERE pais = 'Brasil'
);

-- Exemplo 1: Mesma coisa do de cima, so que sem realizar sub
SELECT nome, sobrenome FROM funcionarios, escritorio AS e
WHERE id_escritorio = e.id AND e.pais = 'Brasil';


-- Exemplo 2:
SELECT f.nome, f.sobrenome, e.pais, p.salario
    FROM pagamentos AS P, funcionarios AS f, escritorio AS e
    WHERE f.id_escritorio = e.id
    AND f.id = p.id_funcionario
    AND salario = (SELECT MAX(salario) FROM pagamentos);

-- Exemplo 3:
SELECT f.nome, f.sobrenome, e.pais, p.salario
    FROM pagamentos AS P, funcionarios AS f, escritorio AS e
    WHERE f.id_escritorio = e.id
    AND f.id = p.id_funcionario
    AND salario < (SELECT AVG(salario) FROM pagamentos);
```

ATUALIZANDO E EXCLUINDO DADOS

```
-- Atualizando dados
#UPDATE
UPDATE atores SET nome = 'Bread Pitt Silva' WHERE id  = 1;

-- Excluindo dados
# DELETE
DELETE FROM genero WHERE id = 3;
```

SQL AVANÇADO

OUTROS OPERADORES

1. `LIMIT e OFFSET`:
    - `LIMIT` : Pestringir o número de linhas que uma consulta SQL retorna. Ele é útil quando você quer ver apenas uma parte dos resultados, como as 10 primeiras linhas.
    - `OFFSET` : Pula um certo número de linhas antes de começar a exibir os resultados. Isso é útil em paginação, como quando você quer começar a ver os resultados a partir da 11ª linha.
        - **Exemplo**: Se o usuário colocar na página 15, ele não vai carregar todo o bloco de dados (páginas anteriores) de uma vez só , ele só carrega a página 15

```
SELECT * FROM produtos
LIMIT 10;  -- Mostra apenas as primeiras 10 linhas


SELECT * FROM produtos
LIMIT 10 OFFSET 10;  -- Pula as 10 primeiras linhas e mostra a



#Juntando os 2
SELECT * FROM produtos
LIMIT 10 OFFSET 10;  -- Pula os primeiros 10 resultados e mos
```

2. Consultas Aninhadas

- É uma subconsulta (consulta dentro de outra consulta)

Exemplo:

```
-- Exemplo 1: Seleciona os nomes dos produtos cujo preço é
maior que a média dos preços
SELECT nome FROM produtos
WHERE preco > (SELECT AVG(preco) FROM produtos);
```

```
-- Exemplo 2: Seleciona o nome dos funcionários e a contage
m de seus dependentes
SELECT f.pnome,
       (
           SELECT COUNT(*)
           FROM dependente AS d
           WHERE d.fcpf = f.cpf
       ) AS numero_dependentes
FROM funcionario AS f;



-- Exemplo 3: Seleciona os produtos cujo preço é maior do q
ue a média de preço da categoria correspondente
SELECT p.nome
FROM produtos AS p
WHERE p.preco > (
    SELECT AVG(p2.preco)
    FROM produtos AS p2
    WHERE p2.categoria_id = p.categoria_id
);


-- Exemplo 4:
SELECT p.nome
FROM personagens P
WHERE p.casa IN (SELECT ID FROM casa WHERE nome='Grifinóri
a')
```

3. Operadores IN e NOT IN

Conceito:

- `IN` filtrar registros que correspondem a qualquer um dos valores dentro de uma lista.

- `NOT IN` exclui registros que correspondem a qualquer um dos valores dentro de uma lista.

Exemplo:

```
-- Exemplo 1: Produtos que pertencem às categorias 1, 2 ou
3
SELECT nome FROM produtos
WHERE categoria_id IN (1, 2, 3);


-- Exemplo 2: Não lista os produtos que pertencem às catego
rias 4 ou 5
SELECT nome FROM produtos
WHERE categoria_id NOT IN (4, 5);


-- Exemplo 3:
SELECT DISTINCT Fcpf
FROM TRABALHA_EM
WHERE (Pnr, Horas) IN (SELECT Pnr, Horas FROM TRABALHA_EM W
HERE Fcpf='12345678');
```

4. Consultas Aninhadas Correlacionadas

Conceito:

- Uma consulta aninhada correlacionada depende de cada linha da consulta externa para ser executada.

Exemplo:

```
sql
Copiar código
-- Seleciona os funcionários cujo salário é maior que a méd
ia do departamento ao qual pertencem
SELECT f.nome FROM funcionarios f
WHERE f.salario > (
```

```
    SELECT AVG(f2.salario)
    FROM funcionarios f2
    WHERE f2.departamento_id = f.departamento_id
);
```

5. Operadores ANY, SOME e ALL

Conceito:

- `ANY` e `SOME` (que são equivalentes) retornam verdadeiro se qualquer valor da subconsulta satisfaz a condição.
- `ALL` retorna verdadeiro se todos os valores da subconsulta satisfazem a condição.

Exemplo:

```
-- Seleciona os produtos que custam mais que qualquer um do
s preços médios por categoria
SELECT nome FROM produtos
WHERE preco > ANY (SELECT AVG(preco) FROM produtos GROUP BY
categoria_id);

-- Seleciona os produtos que custam mais do que todos os pr
eços médios por categoria
SELECT nome FROM produtos
WHERE preco > ALL (SELECT AVG(preco) FROM produtos GROUP BY
categoria_id);
```

6. Operadores EXISTS e NOT EXISTS

Conceito:

- `EXISTS` retorna verdadeiro se a lista tem elementos
- `NOT EXISTS` retorna verdadeiro se é vazio

Exemplo:

```
sql
Copiar código
-- Seleciona os departamentos que têm pelo menos um funcion
ário
SELECT nome FROM departamentos d
WHERE EXISTS (
    SELECT 1 FROM funcionarios f
    WHERE f.departamento_id = d.id
);

-- Seleciona os departamentos que não têm nenhum funcionári
o
SELECT nome FROM departamentos d
WHERE NOT EXISTS (
    SELECT FROM funcionarios f
    WHERE f.departamento_id = d.id
);


-- Encontre personagens que pertencem a 'Corvinal' e usam p
elo menos uma habilidade
SELECT nome
FROM Personagens AS p
WHERE EXISTS (
    SELECT FROM Habilidade AS h
    WHERE p.id = h.id

);
```

LIKE

- `LIKE` *usado para pesquisar valores baseados em texto*

```
#todas as strings que começam com a letra A
SELECT * FROM robots WHERE name LIKE 'A%'
```

```
#todas as strings que terminam com a letra A
SELECT * FROM robots WHERE name LIKE '%A'

#todas as strings que estão em qualquer posição com a letra A
SELECT * FROM robots WHERE name LIKE '%A%'

#1 letra A e o resto pode ser qualquer coisa
SELECT * FROM robots WHERE name LIKE 'A_'

#primeiro seja qualquer um e o ultimo seja A
SELECT * FROM robots WHERE name LIKE '_A'

#2 letra seja A
SELECT * FROM robots WHERE name LIKE '_A_'

#letra A na penultima posição, e o ultiumo caractere seja qua
SELECT * FROM robots WHERE name LIKE '%A_'

#letra A na 2 posição , sendo o primeiro caractere qualquer o
SELECT * FROM robots WHERE name LIKE '_A%'

#strings com 3 caracteres
SELECT * FROM robots WHERE name LIKE '___'

#pelo menos 2 caracteres
SELECT * FROM robots WHERE name LIKE '__%'
```

SUBSTR

- `SUBSTR` *procurar a substring de um valor dado*
  - *Ex: Se você armazenou no formato "cidade, estado" e você queira pegar apenas o estado*

```
SUBSTR(column_name, index, number_of_characters)
```

Exemplo:

```
SELECT * FROM robots WHERE SUBSTR(name, -4) LIKE '20__';
```

LIDANDO COM VIOLAÇÃO DA INTEGRIDADE

1. `No action, restrict`
    - Se você tentar excluir um cliente que tem pedidos, o banco de dados vai impedir a exclusão.

Exemplo

```
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,  -- Tabela de clientes
    name VARCHAR(100)
);

CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,     -- Tabela de pedidos
    customer_id INTEGER REFERENCES customers(customer_id)
    ON DELETE RESTRICT               -- Impede a exclusão do
);
```

2. `ON delete cascade`
    - Quando você apaga um cliente, todos os pedidos dele também são automaticamente apagados.

Exemplo

```
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    name VARCHAR(100)
```

```
);

CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES customers(customer_id)
    ON DELETE CASCADE,  -- Apaga os pedidos do cliente quando
    order_date DATE
);
```

3. `SET NULL e SET DEFAULT`

   - `SET NULL`: Se um gerente sai da empresa e é removido, o projeto dele continua existindo, mas sem um gerente atribuído. Fica como "nenhum gerente".
   - `SET DEFAULT`: Define um valor padrão quando algo é apagado, como atribuir um gerente padrão se o atual é removido.

Exemplo com `SET NULL`:

```
CREATE TABLE Project (
    project_id SERIAL PRIMARY KEY,
    project_name VARCHAR(100),
    manager_id INTEGER REFERENCES Employee(employee_id)
    ON DELETE SET NULL  -- Define o gerente como nulo se ele
);
```

Exemplo com `SET DEFAULT`:

```
- Tabela de funcionários (empregados)
CREATE TABLE Employee (
employee_id SERIAL PRIMARY KEY,
name VARCHAR(100)
);
```

```
- Tabela de projetos com referência ao gerente
CREATE TABLE Project (
project_id SERIAL PRIMARY KEY,
project_name VARCHAR(100),
manager_id INTEGER DEFAULT 1 REFERENCES Employee(employee_id)
ON DELETE SET DEFAULT -- Define o gerente como um valor padrã
);
```