INDICE

==========

como abrir um banco especifico

Abrir o terminal

```
sudo -u postgres psql
```

2° CRIANDO UM BANCO

2. Conecte-se ao PostgreSQL:

Para criar um banco de dados, você precisa estar conectado ao PostgreSQL como um usuário com permissões para criar bancos de dados. Por padrão, o PostgreSQL cria um superusuário chamado "postgres". Você pode se conectar a ele usando o comando `psql` no terminal:

```
sudo -u postgres psql
```

Isso abrirá o shell interativo do PostgreSQL.

3. Crie um banco de dados:
Uma vez conectado ao PostgreSQL, você pode criar um banco de dados usando o comando SQL `CREATE DATABASE`. Por exemplo, para criar um banco de dados chamado "meubanco", você pode fazer o seguinte:

```
CREATE DATABASE meubanco;
```

Certifique-se de terminar os comandos SQL com ponto e vírgula.

4. Saia do shell do PostgreSQL:
Após criar o banco de dados, você pode sair do shell do PostgreSQL digitando:

```
\q
```

Isso o levará de volta ao prompt do terminal.

Agora você criou com sucesso um banco de dados no PostgreSQL em sua máquina Linux. Você pode usar este banco de dados para armazenar e gerenciar seus dados. Lembre-se de que você também pode criar usuários e conceder permissões específicas a eles, dependendo dos requisitos de segurança do seu sistema. Certifique-se de ajustar as configurações de segurança conforme necessário para proteger seus dados.

*/

/*
Lembre-se de que é importante manter o phpMyAdmin atualizado para garantir a segurança do seu sistema. Você pode fazer isso regularmente usando o comando sudo apt update && sudo apt upgrade para atualizar todos os pacotes, incluindo o phpMyAdmin.
*/

OBS: pode ser um banco MySQK tambem

3° Removendo banco

-> Podemos tambem remover os bancos, ou seja deleta-los do sistema;

-> Isso fara com que todos os dados e tabelas sejam perdidos, então tome cuidado
-> comando para deletar o banco de dados é:
        DROP DATABASE nomedobanco;

4° Principais tipos de dados

-> Os tipos de dados do banco funcionam como os tipos de dados de variaveis, em varios niveis para melhor performace;
        * VARCHAR: Texto de 0 a 65535 caracters;
        * TEXT: texto com no maximo 65535 bytes
        * INT: numeros inteiros
        * BIGINT: numeros com maior proporção de int;
        * DATE: data no formato YYYY-MM-DD

5° CRIANDO TABELAS

1. Abra o shell interativo do PostgreSQL se ainda não estiver conectado:

   ```
   sudo -u postgres psql
   ```

2. Agora que você está no shell do PostgreSQL, você pode se conectar ao banco de dados "estudophp" usando o seguinte comando:

   ```sql
   \c estudophp
   ```

   Isso irá conectá-lo ao banco de dados "estudophp". Certifique-se de que não haja erros de digitação no nome do banco de dados.

3. Depois de se conectar ao banco de dados, você pode começar a executar comandos SQL para criar tabelas, inserir dados e realizar outras operações. Por exemplo, para criar uma tabela, você pode usar um comando SQL como este:

   ```sql
   CREATE TABLE minha_tabela (
       id serial PRIMARY KEY,
       nome VARCHAR(255),
       idade INT
   );
   ```

   Este é apenas um exemplo de comando para criar uma tabela simples chamada "minha_tabela" com três colunas: "id", "nome" e "idade". Você pode adaptar esse comando de acordo com suas necessidades.

4. Para sair do shell do PostgreSQL e retornar ao prompt do terminal, use o comando `\q`:

```
\q
```

Agora você está conectado ao banco de dados "estudophp" e pode começar a criar tabelas e executar outras operações SQL nele. Certifique-se de substituir o exemplo acima com os comandos SQL específicos que você deseja executar em seu banco de dados.

6° CRIANDO TABELAS2

No MySQL para criar uma tabela em um banco ja preexistente vamos:

1° vamos abrir o mysql com o comando -> mysql -u cleandro -p
2° logo após vamos abrir o banco com -> USE estudophp;
3° Basta dar apenas este comando:

```
CREATE TABLE outra_nova_tabela(tipo VARCHAR(100) descricao TEXT,
ano_fabricacao INT);
```

7° Removendo Tabelas

-> Há possibilidade de remover tabelas do banco;
-> Os dados são removidos para sempre;
-> O comando paara deletar é:

```
DROP TABLE nome_do_banco;
```

7° Removendo Tabelas

Para alterar uma tabela no PostgreSQL, você pode usar o comando SQL `ALTER TABLE`. O comando `ALTER TABLE` permite adicionar, modificar ou excluir colunas, bem como fazer outras alterações na estrutura de uma tabela existente. Aqui estão alguns exemplos de como usar o `ALTER TABLE` para realizar diferentes tipos de alterações:

1. **Adicionar uma nova coluna**:

   Para adicionar uma nova coluna a uma tabela existente, use o seguinte comando:

   ```sql
   ALTER TABLE nome_da_tabela
   ADD nome_da_coluna tipo_de_dado;
   ```

   Por exemplo, para adicionar uma coluna chamada "email" do tipo VARCHAR(255) a uma tabela chamada "usuarios", você pode fazer o seguinte:

   ```sql
   ALTER TABLE usuarios
   ADD email VARCHAR(255);
   ```

'
2. **Modificar uma coluna existente**:

   Para modificar uma coluna existente, use o comando `ALTER TABLE` com a cláusula `ALTER COLUMN`. Por exemplo, para alterar o tipo de dados de uma coluna de "int" para "bigint" em uma tabela chamada "minha_tabela", você pode fazer o seguinte:

   ```sql
   ALTER TABLE minha_tabela
   ALTER COLUMN nome_da_coluna TYPE bigint;
   ```

3. **Renomear uma coluna**:

   Para renomear uma coluna existente, use a cláusula `RENAME COLUMN` no comando `ALTER TABLE`. Por exemplo, para renomear a coluna "nome" para "nome_completo" em uma tabela chamada "usuarios", você pode fazer o seguinte:

   ```sql
   ALTER TABLE usuarios
   RENAME COLUMN nome TO nome_completo;
   ```

4. **Excluir uma coluna**:

   Para excluir uma coluna de uma tabela, use a cláusula `DROP COLUMN` no comando `ALTER TABLE`. Por exemplo, para excluir a coluna "telefone" de uma tabela chamada "contatos", você pode fazer o seguinte:

   ```sql
   ALTER TABLE contatos
   DROP COLUMN telefone;
   ```

Lembre-se de que, ao fazer alterações em uma tabela, você pode perder dados existentes se não tomar cuidado. Portanto, é recomendável fazer backup dos dados antes de realizar alterações significativas em uma tabela. Além disso, certifique-se de que a estrutura da tabela esteja de acordo com seus requisitos de aplicação após a conclusão das alterações.

8° abrindo um banco especifico

Para visualizar os dados armazenados dentro do banco de dados "estudophp" no PostgreSQL, você pode usar consultas SQL para recuperar informações das tabelas. Aqui estão os passos básicos para fazer isso:

1. **Abra o shell interativo do PostgreSQL** se você ainda não estiver conectado:

   ```
   sudo -u postgres psql
```

```

2. **Conecte-se ao banco de dados "estudophp"** usando o comando `\c`:

   ```sql
   \c estudophp;
   ```

3. **Execute consultas SQL para visualizar os dados**. Por exemplo, para listar todos os registros de uma tabela chamada "usuarios", você pode usar a seguinte consulta SQL:

   ```sql
   SELECT * FROM usuarios;
   ```

   Esta consulta irá recuperar todos os registros da tabela "usuarios" e exibir seus dados.

4. Você pode usar várias consultas SQL para visualizar e manipular os dados da tabela de acordo com suas necessidades. Por exemplo, você pode usar a cláusula `WHERE` para filtrar resultados ou realizar operações de agregação, como `COUNT`, `SUM`, `AVG`, etc., para resumir os dados.

5. Para sair do shell do PostgreSQL e retornar ao prompt do terminal, use o comando `\q`:

   ```
   \q
   ```

Lembre-se de que a visualização de dados no banco de dados depende da estrutura das tabelas e das consultas SQL que você executa. Certifique-se de adaptar suas consultas de acordo com as tabelas e os dados que você deseja visualizar.

->OBS: Para ver o que tem dentro do banco estudophp basta digitar \dt

---

-verificando o conteudo da minha tabela-

Para verificar o conteúdo de uma tabela no PostgreSQL, você pode usar a consulta SQL `SELECT`. Aqui está como você pode consultar os dados de uma tabela chamada "minha_tabela" no esquema "public":

```sql
SELECT * FROM minha_tabela;
```

Este comando SQL irá selecionar todos os registros da tabela "minha_tabela" e retorná-los como resultado da consulta.

Para executar esta consulta:
```

1. Abra o shell interativo do PostgreSQL, se ainda não estiver conectado:

   ```
   sudo -u postgres psql
   ```

2. Conecte-se ao banco de dados em que a tabela está localizada, por exemplo, "estudophp":

   ```sql
   \c estudophp
   ```

3. Execute a consulta SQL para visualizar os dados da tabela:

   ```sql
   SELECT * FROM minha_tabela;
   ```

Isso irá listar todos os registros da tabela "minha_tabela" e exibir suas informações no terminal. Certifique-se de que a tabela "minha_tabela" está no esquema "public" e que você está conectado ao banco de dados correto antes de executar a consulta. Se houver dados na tabela, eles serão exibidos como saída da consulta.

->REMOVENDO COLUNA DE DENTRO DA TABELA<-

Para verificar o conteúdo de uma tabela no PostgreSQL, você pode usar a consulta SQL `SELECT`. Aqui está como você pode consultar os dados de uma tabela chamada "minha_tabela" no esquema "public":

```sql
SELECT * FROM minha_tabela;
```

Este comando SQL irá selecionar todos os registros da tabela "minha_tabela" e retorná-los como resultado da consulta.

Para executar esta consulta:

1. Abra o shell interativo do PostgreSQL, se ainda não estiver conectado:

   ```
   sudo -u postgres psql
   ```

2. Conecte-se ao banco de dados em que a tabela está localizada, por exemplo, "estudophp":

   ```sql
   \c estudophp
   ```

3. Execute a consulta SQL para visualizar os dados da tabela:

```sql
SELECT * FROM minha_tabela;
```

Isso irá listar todos os registros da tabela "minha_tabela" e exibir suas informações no terminal. Certifique-se de que a tabela "minha_tabela" está no esquema "public" e que você está conectado ao banco de dados correto antes de executar a consulta. Se houver dados na tabela, eles serão exibidos como saída da consulta.

-> OBS: Para modificar o a quantidade de dados que uma coluna recebe podemos utilizar por exemplo:

```
ALTER TABLE carros
MODIFY COLUMN descrição VARCHAR(500);
```

10° inserindo dados dentro da tabela

```
INSERT INTO not_null (nome, idade)
VALUES ('João', 30);
```

Para inserir dados em uma tabela no PostgreSQL, você pode usar o comando SQL `INSERT INTO`. Supondo que você queira inserir dados na tabela "not_null", aqui está como você pode fazer isso:

```sql
INSERT INTO not_null (coluna1, coluna2, coluna3)
VALUES (valor1, valor2, valor3);
```

Substitua "not_null" pelo nome da sua tabela e especifique os nomes das colunas (coluna1, coluna2, coluna3) nas quais você deseja inserir valores. Em seguida, forneça os valores correspondentes (valor1, valor2, valor3) na mesma ordem das colunas.

Aqui está um exemplo concreto. Suponhamos que a tabela "not_null" tenha as colunas "nome" e "idade", e você deseja inserir um novo registro com os valores "João" e 30:

```sql
INSERT INTO not_null (nome, idade)
VALUES ('João', 30);
```

Após executar este comando, um novo registro será inserido na tabela "not_null" com os valores especificados.

Certifique-se de que os valores que você está inserindo estejam no formato correto e cumpram quaisquer restrições de dados definidas para as colunas (por exemplo, restrições de tipo de dados ou restrições NOT NULL).

Após inserir os dados, você pode verificar se eles foram inseridos com sucesso usando uma consulta SELECT:

```sql
SELECT * FROM not_null;
```

Isso exibirá todos os registros na tabela "not_null", incluindo o que você acabou de inserir.

11° validação de dados unicos no banco

No PostgreSQL, você pode aplicar uma restrição de chave única a uma ou mais colunas de uma tabela para garantir que os valores nessas colunas sejam únicos, ou seja, não haja duplicatas. Isso é útil quando você deseja evitar que dados duplicados sejam inseridos em uma tabela. Aqui está um exemplo de como criar uma restrição de chave única em uma tabela:

Suponha que você tenha uma tabela chamada "usuarios" e deseje garantir que o campo "email" seja único para cada usuário. Você pode criar uma restrição de chave única da seguinte maneira:

```sql
CREATE TABLE usuarios (
    id serial PRIMARY KEY,
    nome VARCHAR(255),
    email VARCHAR(255) UNIQUE, -- Restrição de chave única
    senha VARCHAR(255)
);
```

No exemplo acima, a coluna "email" tem uma restrição de chave única, especificada pela palavra-chave `UNIQUE`. Isso garante que nenhum valor duplicado seja inserido na coluna "email". Se alguém tentar inserir um registro com um valor de "email" que já existe na tabela, o PostgreSQL lançará um erro e impedirá a inserção.

Aqui estão alguns exemplos de como essa restrição funciona:

Inserindo um novo usuário com um email único:

```sql
INSERT INTO usuarios (nome, email, senha) VALUES ('Alice',
'alice@example.com', 'senha123');
```

Inserindo um novo usuário com um email que já existe na tabela (irá falhar):

```sql
```

```
INSERT INTO usuarios (nome, email, senha) VALUES ('Bob',
'alice@example.com', 'outrasenha');
-- Isso resultará em um erro de violação de chave única
```

Você também pode aplicar restrições de chave única a várias colunas, o que garante que a combinação de valores nessas colunas seja única.

Por exemplo, se você quisesse garantir que o par de "nome" e "email" fosse único, você poderia definir uma restrição de chave única da seguinte maneira:

```sql
CREATE TABLE usuarios (
    id serial PRIMARY KEY,
    nome VARCHAR(255),
    email VARCHAR(255),
    senha VARCHAR(255),
    UNIQUE (nome, email) -- Restrição de chave única em nome e email
);
```

Isso garantiria que nenhum par de "nome" e "email" duplicado seja inserido na tabela "usuarios".

9° verificando o que tem dentro da minha tabela

Para verificar o conteúdo de uma tabela no PostgreSQL, você pode usar a consulta SQL SELECT. Aqui está como você pode consultar os dados de uma tabela chamada "minha_tabela" no esquema "public":

```
SELECT * FROM minha_tabela;
```

---

12 ° comando para verificar as parametrizações do php

```
<?php
phpinfo();
?>
```

Basta apenas colocar para rodar este comando no navegador, vai abrir uma pagina com todas as configurações do php em sua maquina, onde vc vai poder verificar se esta faltando algo para ser habilitado.

13° comando para dar start no mysql

```
sudo systemctl start mysql
```

este comando serve para caso o serviço do mysql esteja parado em sua maquina

---

14° comando para verificar o status do mysql

```
sudo systemctl status mysql
```

15° como reiniciar o mysql

sudo service mysql restart

16° para entrar no mysql

mysql -u cleandro -p

17° para verificar banco criado

SHOW DATABASES LIKE 'estudophp';

18° para colocar algo dentro do banco

Abrir o mysql com
e depois rodar o comando por exemplo: USE estudophp;

19° para verificar o que tem dentro de um banco especifico

1° com o mysql aberto insira o seguinte comando "mysql -u root USE estudophp;"
2° em seguida " mysql -u root USE estudophp; "
-> vais receber uma messagem parecida com esta

```
+----------------------+
| Tables_in_estudophp |
+----------------------+
| testeando           |
+----------------------+
```

Neste caso tenho o banco "estudophp" temos uma tabela dentro por nome "testando"

20° como chamar um banco em especifico

USE estudophp;

21° ver tabela dentro de um banco especifico "estudophp"

Para visualizar as tabelas dentro de um banco de dados no MySQL, você pode usar o comando `SHOW TABLES`. No seu caso, para ver a tabela "pessoas" dentro do banco de dados "estudophp", siga estas etapas:

1. Primeiro, certifique-se de que você já está conectado ao MySQL e selecionou o banco de dados "estudophp". Se você não estiver conectado ou no banco de dados correto, pode fazer o seguinte:

   ```sql
   USE estudophp;
   ```

2. Agora, para listar todas as tabelas no banco de dados "estudophp", use o seguinte comando:

   ```sql
   SHOW TABLES;
   ```

Isso retornará uma lista de todas as tabelas no banco de dados "estudophp", incluindo a tabela "pessoas", se ela existir. Você verá algo parecido com:

```
+--------------------+
| Tables_in_estudophp |
+--------------------+
| pessoas            |
| outra_tabela       |
| ...                |
+--------------------+
```

Neste exemplo, você verá a tabela "pessoas" na lista.

OBS: para inserir uma tabela dentro do banco sempre dar um USE antes

---

22° exemplo de tabela para cadastro

Exemplo de cadastro com os campos nome completo, data de nascimento, endereço, telefone e email.

```sql
USE estudophp;

CREATE TABLE IF NOT EXISTS clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome_completo VARCHAR(255) NOT NULL,
    data_nascimento DATE,
    endereco VARCHAR(255),
    telefone VARCHAR(15),
```

```
    email VARCHAR(255)
);
```

Neste exemplo:

- Estamos usando a instrução `CREATE TABLE` para criar uma tabela chamada "clientes" no banco de dados "estudophp".
- A coluna "id" é definida como um campo de identificação (primary key) com a opção "AUTO_INCREMENT", o que significa que ela será preenchida automaticamente com valores incrementais únicos para cada registro.
- A coluna "nome_completo" é do tipo VARCHAR e é definida como "NOT NULL", o que significa que não pode ser deixada em branco.
- A coluna "data_nascimento" é do tipo DATE para armazenar a data de nascimento.
- As colunas "endereco", "telefone" e "email" são do tipo VARCHAR para armazenar informações de endereço, telefone e email.

Depois de executar essa instrução SQL, você terá uma tabela "clientes" pronta para armazenar esses dados no banco de dados "estudophp". Você pode adaptar essas colunas e tipos de dados conforme necessário para atender aos requisitos do seu projeto.

23° verificando itens que estao em uma tabela especifica

```
USE estudophp;

SELECT * FROM clientes;
```

24° como alterar uma tabela

Com o mysql rodando e o banco ativo que desja aleterar alguma tabela basta apenas digitar o comando:

```
ALTER TABLE carros
-> ADD motor INT;
```

Nesta banco por nome 'carros' agora temos uma coluna por nome motor

---

25° verificando as colunas dentro das tabelas

```
USE estudophp;
DESCRIBE carros;
```

Vai ficar mais ou menos assim no terminal

```
mysql> USE estudophp;
Database changed
mysql> DESCRIBE carros;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
```

```
| marca          | varchar(100) | YES  |     | NULL    |       |
| descricao      | text         | YES  |     | NULL    |       |
| ano_fabricacao | int          | YES  |     | NULL    |       |
| motor          | int          | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

Bom dia! Para ver o que está dentro da tabela "carros" no MySQL, você pode usar a instrução SELECT. Suponhamos que você queira listar todos os dados da tabela "carros", você pode fazer o seguinte:

```sql
SELECT * FROM carros;
```

Isso irá selecionar todas as colunas e todas as linhas da tabela "carros" e exibir os resultados.

Se você quiser selecionar apenas algumas colunas específicas, você pode listar essas colunas no lugar do asterisco (*). Por exemplo, se você quiser ver apenas as colunas "modelo" e "ano", você pode fazer o seguinte:

```sql
SELECT modelo, ano FROM carros;
```

Lembre-se de que você deve estar conectado ao banco de dados onde a tabela "carros" está localizada quando executar essas consultas. Certifique-se de ter selecionado o banco de dados correto usando o comando "USE" antes de executar a consulta, se necessário.

26° como remover uma coluna de uma tabela

```
mysql> ALTER TABLE carros
-> DROP ano_fabricacao;
```

27° como modificar uma coluna
(modificando o tipo da coluna)

```
ALTER TABLE carros
MODIFY COLUMN descricao VARCHAR(500);
```

-> digite este comando para verificar o tipo de dados que tem dentro de uma tabela especifica:

```
mysql> DESCRIBE carros;
```

Antes o banco estava assim:

```
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
```

```
+----------------+--------------+------+-----+---------+-------+
| marca          | varchar(100) | YES  |     | NULL    |       |
| descricao      | text         | YES  |     | NULL    |       |
| motor          | int          | YES  |     | NULL    |       |
| vidro_eletrico | int          | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

Depois o banco ficou assim:

```
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| marca          | varchar(100) | YES  |     | NULL    |       |
| descricao      | varchar(500) | YES  |     | NULL    |       |
| motor          | int          | YES  |     | NULL    |       |
| vidro_eletrico | int          | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

28° inserindo camadas de verificação (Not NULL)
[Impedir que o dado seja mandado vazio]

```
CREATE TABLE notnull(
    nome VARCHAR(100) NOT NULL,
    idade INT
);
```

-> Esta tabela foi criada dentro do banco estudo PHP
-> Na coluna nome tem um valor NOT NULL, o que siguinifica dizer que o usoario não vai poder preecher este formulario e deixar o campo nome vazio

---

inserido valores dentro das colunas da tabebelas

```
INSERT INTO notnull (nome, idade) VALUES ("mateus", 29)
```

-> Nesta tabela recebe dois dados de preechimento => Nome e idade;

29° UNIQUE - Evitand dados duplicadodos

```
CREATE TABLE mailmarkting(
      email VARCHAR(255) NOT NULL UNIQUE,
      nome VARCHAR(255)
      );
```

-Aqui neste codigo onde crio uma nova tabela dentro do banco estudophp no campo email esta sendo impedido atraves do "NOT NULL" e "UNIQUE" a inserção pelo usuario de valores vazios ou duplicados

-> Para fazer o teste para verificação do bloqueio podemos inserir dados na tabela mailmarkting

=> 30° inserindo dados na coluna mailmarketing

```
INSERT INTO mailmarketing (gmail, nome) VALUES
("cleandrosetta@gmail.com", "jorge");
```

31° como colocar um valor unico

```
INSERT INTO itens(nome, descricao) VALUES ('abajur', 'em mais ou menos
execelente estado');
```

32° deletando itnes pelo id

```
DELETE FROM instens WHERE id = 2;
```

33° deletando itens pelo nome

```
DELETE FROM instens WHERE nome = 'VALOR_A_SERdeleTADO';
```

34° SELECIONANDO DADOS_1

```
SELECT * FROM tabela; (aqui eu estu trazendo tudo que tem dentro desta
tablea "itens")
```

Selecionamdo dados especificos (buscando apenas as culunas)

```
-tras todos os nomes:
mysql> SELECT nome FROM itens;
+------------------+
| nome             |
+------------------+
| abajur           |
| Carrinho de mao  |
| DRONE            |
| TECLADO          |
| Mouse            |
| Microfone        |
+------------------+
```

35° SELECIONANDO DADOS_2 Que começam com a letra 'a'

```
SELECT nome FROM itens WHERE nome LIKE 'a%';
```

este codigo vai fazer uma busca de todos os nomes que começam com a letra 'a':

```
mysql> SELECT nome FROM itens WHERE nome LIKE 'a%';
+--------+
| nome   |
+--------+
| abajur |
+--------+
```

```
1 row in set (0,00 sec)
```

36° SELECIONANDO DADOS_3 Que contem um nome especifico

```
SELECT nome FROM itens WHERE nome = 'drone';
```

37° buscando itens que sao maior que 3

```
SELECT nome FROM itens WHERE id > 3;
```

38° trazendo duas colunas o 'id' e o 'nome'

```
mysql> SELECT id, nome FROM itens WHERE id <= 3;
+----+-----------------+
| id | nome            |
+----+-----------------+
|  1 | abajur          |
|  3 | Carrinho de mao |
+----+-----------------+
2 rows in set (0,00 sec)
```

39° ORDEBY

-com este codigo podemos verificar itens que forma adcionados fazendo uma busca dos primeiros ou dos ultimos

-> Neste exemplo dentro do banco "estudophp" na tabela "itens" estamos buscando todos os ultimos itens que foram acrescentados, ou seja tras do ultimo para o primeiro.

```
mysql> SELECT * FROM itens ORDER BY id DESC;
+----+-----------------+-------------------------------------------------+
| id | nome            | descricao                                       |
+----+-----------------+-------------------------------------------------+
|  8 | Microfone       | Microfone sem marca, muito barato               |
|  7 | Mouse           | O muse esta usado e precisando de reparos       |
|  6 | TECLADO         | Teclado da microsoft                            |
|  5 | DRONE           | VOA 3KM DE DISTANCIA                            |
|  3 | Carrinho de mao | O produto esta novo                             |
|  1 | abajur          | abajur novo em mais ou menos excelente estado
```

-Trazendo todos os itens que são maior que 2 (POR ORDEM ALFABETICA)

```
mysql> SELECT * FROM itens WHERE id > 2 ORDER BY nome;
+----+-----------------+-----------------------------------------+
```

```
| id | nome            | descricao                                  |
+----+-----------------+--------------------------------------------+
|  3 | Carrinho de mao | O produto esta novo                        |
|  5 | DRONE           | VOA 3KM DE DISTANCIA                       |
|  8 | Microfone       | Microfone sem marca, muito barato          |
|  7 | Mouse           | O muse esta usado e precisando de reparos  |
|  6 | TECLADO         | Teclado da microsoft                       |
+----+-----------------+--------------------------------------------+
```

-Trazendo todos os itens que são maior que 2 ordenando por nome decrescente (POR ORDEM ALFABETICA)

```
mysql> SELECT * FROM itens WHERE id > 2 ORDER BY nome DESC;
+----+-----------------+--------------------------------------------+
| id | nome            | descricao                                  |
+----+-----------------+--------------------------------------------+
|  6 | TECLADO         | Teclado da microsoft                       |
|  7 | Mouse           | O muse esta usado e precisando de reparos  |
|  8 | Microfone       | Microfone sem marca, muito barato          |
|  5 | DRONE           | VOA 3KM DE DISTANCIA                       |
|  3 | Carrinho de mao | O produto esta novo                        |
+----+-----------------+--------------------------------------------+
```

```
+----+-----------------+-----------------------------------------------
```

40° Atualizando dados

OBS: sempre que for atualizar a tabela utilizar o WHERE para não atualizar a tabela por completo acidentalmente

-modificando um item da coluna nome (DRONE)

```
mysql> SELECT * FROM itens;
+----+-----------------+-------------------------------------------------+
| id | nome            | descricao                                       |
+----+-----------------+-------------------------------------------------+
|  1 | abajur          | abajur novo em mais ou menos excelente estado   |
|  3 | Carrinho de mao | O produto esta novo                             |
|  5 | DRONE DJI       | VOA 3KM DE DISTANCIA                            |
|  6 | TECLADO         | Teclado da microsoft                            |
|  7 | Mouse           | O muse esta usado e precisando de reparos       |
|  8 | Microfone       | Microfone sem marca, muito barato               |
+----+-----------------+-------------------------------------------------+
```

modificando um iten da coluna descricao (VOA 3KM DE DISTANCIA -> para-> voa 10km de distandia)

-> Podemos atualizer mais de uma coluna por vez

```
mysql> UPDATE itens
    -> SET nome = "Mouse made in BRASIL",
    -> descricao = "Mouse feito no Brasil, referencia mondial"
    -> WHERE id = 7;
```

```
+----+----------------------+-------------------------------------------
---+
| id | nome                 | descricao
|
+----+----------------------+-------------------------------------------
---+
|  1 | abajur               | abajur novo em mais ou menos excelente
estado |
|  3 | Carrinho de mao      | O produto esta novo
|
|  5 | DRONE DJI            | voa 10km de distancia
|
|  6 | TECLADO              | Teclado da microsoft
|
|  7 | Mouse made in BRASIL | Mouse feito no Brasil, referencia mondial
|
|  8 | Microfone            | Microfone sem marca, muito barato
|
+----+----------------------+-------------------------------------------
-------------------------+
```

41° deletando dados

OBS: sempre que usar o DELETE utilizar o WHERE para não deletar todos os dados acidentalmente.

```
mysql> DELETE FROM itens WHERE id = 7;
```

```
+-----------------------------------------------------------------------+
```

42° Criando usuarios

```
CREAT USER 'teste'@'localhost' INDENTIFIED BY 'teste123';
GRANT ALL PRIVILEGES ON *.* TO 'teste'@'localhost';
FLUSH PRIVILEGES;
```