



# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 11 – SQL II

# O QUE IREMOS APRENDER

**01**

RESUMO DA AULA PASSADA

**02**

CONTEXTUALIZAÇÃO DA AULA DE HOJE

**03**

COMANDOS SQL

**04**

MÃOS NO CÓDIGO

# Resumo da aula passada

SQL é uma linguagem usada para interagir com bancos de dados, enquanto um banco de dados é uma estrutura para armazenar dados de forma organizada. MySQL é um sistema de gerenciamento de banco de dados relacional que utiliza SQL como linguagem principal para interação com os dados. Os comandos SQL permitem a criação, manipulação e recuperação de informações em um banco de dados MySQL.



**Banco de Dados**

# Resumo da aula passada

Na aula passada também abordamos alguns comando SQL, ou seja, algumas manipulações de dados.

**CREATE TABLE:** Usado para criar uma nova tabela em um banco de dados.

**CREATE DATABASE:** Usado para criar um novo banco de dados no sistema.

**USE DATABASE:** Usado para selecionar um banco de dados específico com o qual você deseja trabalhar.

# Resumo da aula passada

**DROP TABLE:** Usado para excluir uma tabela inteira e todos os seus dados. Esta ação é irreversível.

**TRUNCATE TABLE:** Usado para excluir todos os dados de uma tabela, mantendo a estrutura da tabela intacta.



# Contextualização da aula de hoje

A manipulação de dados em SQL envolve a criação, atualização, exclusão e recuperação de informações em um banco de dados relacional. A manipulação de dados em SQL é essencial para criar, atualizar e gerenciar informações em um banco de dados relacional. Esses comandos permitem que você interaja de forma eficiente com os dados, garantindo a integridade e a consistência das informações.

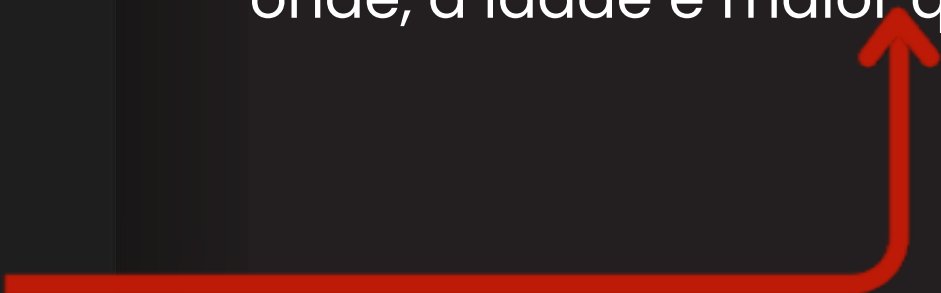


# Comandos SQL - INSERT

Usado para adicionar novos registros a uma tabela. Você que fornece os valores deseja inserir nas colunas correspondentes.

```
1  INSERT INTO alunos (nome, idade) VALUES ("Maria Silva", 22);
2
3  INSERT INTO alunos (nome, idade) VALUES
4  ("Maria Silva", 22),
5  ("Ana Rodrigues", 21),
6  ("Pedro Lima", 23);
7
8  INSERT INTO alunos (nome, idade)
9  SELECT nome, idade FROM alunos_antigos WHERE idade > 20;
10
```

Este comando insere dados na tabela "alunos" com base em uma consulta SELECT em outra tabela "alunos\_antigos" WHERE - onde, a idade é maior que 20.



# Comandos SQL - SELECT

Usado para recuperar dados de uma ou mais tabelas em um banco de dados. Você pode filtrar os resultados com a cláusula WHERE e ordená-los com a cláusula ORDER BY.



```
1 SELECT nome FROM clientes;
```

Neste exemplo, o comando SELECT é usado para recuperar dados da coluna "nome" da tabela "clientes". O resultado será uma lista de nomes de todos os clientes armazenados na tabela.




```
1 SELECT nome, idade FROM alunos WHERE idade < 20;  
2 SELECT * FROM alunos WHERE cidade = "São Paulo";  
3 SELECT * FROM alunos WHERE curso = "Ciência da Computação" AND cidade = "Rio de Janeiro";  
4 SELECT DISTINCT cidade FROM alunos;
```



# Comandos SQL - UPDATE

Usado para modificar dados existentes em uma tabela. Você especifica quais registros devem ser atualizados e quais valores devem ser atribuídos a eles.




```
1 UPDATE alunos
2 SET idade = 23
3 WHERE nome = "Maria Silva";
```

Este comando atualiza a idade da aluna "Maria Silva" para 23 anos na tabela "alunos".

# Comandos SQL - DELETE

Usado para excluir registros de uma tabela com base em critérios específicos. Isso remove permanentemente os dados.

Certifique-se de usar a cláusula WHERE para identificar os registros que deseja excluir e esteja ciente de que o comando DELETE é uma operação irreversível que remove permanentemente os dados.



```
1 DELETE FROM alunos;  
2 DELETE FROM alunos  
3 WHERE idade = 23;
```

# Comandos SQL - ALTER TABLE

Usado para modificar a estrutura de uma tabela existente, como adicionar, modificar ou excluir colunas.



```
1 ALTER TABLE alunos  
2 CHANGE COLUMN curso curso_atual varchar(50);
```

Este comando renomeia a coluna "curso" para "curso\_atual" e altera o tipo de dados para VARCHAR(50).



```
1 ALTER TABLE alunos  
2 ADD COLUMN data_matricula DATE;
```

Este comando adiciona uma nova coluna chamada "data\_matricula" à tabela "alunos", que armazenará datas de matrícula.


# Comandos SQL - CONSTRAINTS

Você pode aplicar restrições, como chaves primárias, chaves estrangeiras, restrições únicas e verificação, para garantir a integridade dos dados.



# Comandos SQL - CONSTRAINTS

**Chave Primária (Primary Key):** Uma constraint de chave primária é usada para identificar de forma exclusiva cada registro em uma tabela. Ela garante que os valores em uma coluna (ou um conjunto de colunas) sejam únicos e não nulos.



```
1 CREATE TABLE alunos(  
2     id_aluno INT PRIMARY KEY  
3     nome VARCHAR(50)  
4 )
```



# Comandos SQL - CONSTRAINTS

**Chave Estrangeira (Foreign Key):** Uma constraint de chave estrangeira estabelece uma relação entre duas tabelas, garantindo que os valores em uma coluna correspondam aos valores de outra tabela.



```
1 CREATE TABLE alunos(  
2     id INT PRIMARY KEY  
3     id_aluno INT,  
4     id_curso INT,  
5     FOREIGN KEY (id_aluno) REFERENCES alunos(id_aluno),  
6     FOREIGN KEY (id_curso) REFERENCES cursos(id_curso)  
7 );  
8
```

# Comandos SQL - CONSTRAINTS


**Restrição Única (Unique Constraint):** Uma restrição única garante que os valores em uma coluna sejam únicos, mas não necessariamente que sejam não nulos. Ela pode ser usada para impedir a inserção de duplicatas.



```
1 CREATE TABLE produtos (  
2     codigo_produto INT UNIQUE,  
3     nome_produto VARCHAR(100)  
4 );
```

# Comandos SQL - CONSTRAINTS

**Restrição de Verificação (Check Constraint):** Uma restrição de verificação permite especificar uma condição que os valores em uma coluna devem atender. Isso pode ser usado para garantir que os dados estejam dentro de um intervalo específico.



```
1 CREATE TABLE funcionarios (  
2     id_funcionario INT,  
3     salario DECIMAL(10, 2),  
4     CHECK (salario >= 0)  
5 );
```

# Comandos SQL - CONSTRAINTS

**Restrição de Não Nulo (Not Null Constraint):** Uma restrição de não nulo garante que uma coluna não pode conter valores nulos, ou seja, todos os registros devem ter valores para essa coluna.



```
1 CREATE TABLE clientes (  
2     id_cliente INT,  
3     nome VARCHAR(50) NOT NULL  
4 );
```

# ATIVIDADE PRÁTICA 1

Crie uma tabela "pedidos" com as colunas "id\_pedido", "id\_cliente" e "data\_pedido". Adicione uma constraint de chave estrangeira na coluna "id\_cliente" para garantir que um pedido só possa ser feito por um cliente existente na tabela "clientes".



# ATIVIDADE PRÁTICA 2

Crie uma tabela "produtos" com as colunas "id\_produto", "nome\_produto" e "preco". Adicione uma constraint de verificação para garantir que o preço do produto seja maior que zero.

# ATIVIDADE PRÁTICA 3

Adicione uma constraint de restrição única na coluna "email" da tabela "clientes" para garantir que nenhum cliente possa ter o mesmo endereço de e-mail.

# ATIVIDADE PRÁTICA 4

Escreva uma consulta que recupere o nome de todos os alunos que tenham mais de 20 anos e que estejam matriculados em cursos de "Matemática". Use uma subconsulta para realizar essa tarefa.

# DESAFIO PRÁTICO

## Sistema de gerenciamento de vendas

Você está criando um banco de dados para gerenciar vendas de produtos em uma loja online. Você precisa projetar o esquema do banco de dados e escrever consultas SQL para atender a várias necessidades da loja. Aqui estão os requisitos:

Crie um banco de dados chamado "loja\_online" e as seguintes tabelas:

**Produtos:** Armazena informações sobre produtos, incluindo um ID, nome, preço e quantidade em estoque.

# DESAFIO PRÁTICO

## Sistema de gerenciamento de vendas

**Clientes:** Armazena informações sobre os clientes, incluindo um ID, nome, endereço de e-mail e histórico de compras.

**Pedidos:** Registra informações sobre os pedidos feitos pelos clientes, incluindo um ID, data do pedido, ID do cliente e status do pedido.

**itens\_Pedido:** Registra os produtos incluídos em cada pedido, incluindo um ID, ID do pedido, ID do produto e quantidade.



# DESAFIO PRÁTICO

## Sistema de gerenciamento de vendas

Imagine que você está trabalhando no desenvolvimento de um sistema de gerenciamento de vendas para uma loja. Você já possui algumas tabelas básicas criadas no banco de dados. Agora, você precisa criar consultas SQL para realizar algumas operações específicas. Considere as seguintes tabelas:

Tabela "Clientes":

Colunas: id\_cliente (chave primária), nome\_cliente, email\_cliente.

Tabela "Produtos":

Colunas: id\_produto (chave primária), nome\_produto, preco\_produto. Tabela "Vendas":

Colunas: id\_venda (chave primária), id\_cliente (chave estrangeira referenciando a tabela "Clientes"), data\_venda.



# DESAFIO PRÁTICO

## Sistema de gerenciamento de vendas

- a) Crie uma consulta SQL para selecionar todos os clientes cadastrados.
- b) Escreva um código SQL para inserir um novo produto chamado "Notebook" com o preço de R\$ 2.500,00 na tabela "Produtos".
- c) Atualize o nome do cliente com o id\_cliente igual a 1 para "Maria Silva".
- d) Remova todos os registros da tabela "Vendas" que ocorreram antes de 01 de janeiro de 2023.

# Material Complementar

- Exploração: Não tenha medo de explorar e testar diferentes códigos. A experimentação é uma grande aliada da aprendizagem.
  - Perguntas: Faça perguntas, seja curioso! Entender o "porquê" das coisas ajuda a consolidar o conhecimento.
  - Revisão: Revise o que aprendeu, tente explicar para si mesmo ou para outras pessoas. Ensinar é uma ótima forma de aprender.
- Prática: A prática leva à perfeição. Quanto mais exercícios fizer, mais fácil será lembrar e entender os conceitos.





# SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 12 DE PYTHON:  
SQL CRUD



**INFINITY SCHOOL**  
VISUAL ART CREATIVE CENTER



# SQL III

Em SQL, um **relacionamento** se refere à forma como as tabelas em um banco de dados estão associadas ou conectadas. Os relacionamentos são fundamentais para organizar dados e permitir que você consulte informações de maneira eficaz através de várias tabelas.

```
...event('on' + type, callback);  
  
function decorate(event) {  
  event = event || window.event;  
  var target = event.target || event.srcElement;  
  if (target && (target.getAttribute('action') || target.  
    ga(function (tracker) {  
      var linkerParam = tracker.get('linkerParam');  
      document.cookie = '_shopify_ga=' + linkerParam + '  
    }));  
  
  listener(window, 'load', function(){  
    for (i=0; i < document.forms.length; i++) {  
      if (document.forms[i].getAttribute('action')  
        && (/cart/).test(document.forms[i].getAttribute('action')) {  
        document.forms[i].submit(), decorate();  
      }  
    }  
  })  
}
```





# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 11 – SQL II