# BOSON TREINAMENTOS

**--Criar um banco de dados chamado "db_Biblioteca"**

**-- ON PRIMARY indica a utilização do grupo de arquivos primarios do sql**
**--NO SQL SERVER sempre utilize aspas simples, para digitar textos strings**

```
CREATE DATABASE db_Biblioteca ON PRIMARY (
NAME = db_Biblioteca,  --Nome
FILENAME ='C:\Program Files\Microsoft SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\db_Biblioteca.MDF', --Localização do arquivo
no disco e o nome do arquivo
SIZE = 6MB, --Tamanho do banco de dados, vai começar em 6MB
MAXSIZE=15MB, --Tamanho maximo do banco de dados
FILEGROWTH=10% --O banco de dados vai crescer de 10 em 10 por cento
);
```

**--Deletando um Banco de Dados**

```
DROP DATABASE db_Biblioteca;
```

**--Selecionando o banco de dados**

```
USE db_Biblioteca;
```

**--Comando que informa o tamanho, taxa de crescimento e local do banco de dados**

```
sp_helpdb db_Biblioteca;
```

**--Criando uma tabela**

```
CREATE TABLE tbl_Livro(
ID_Livro SMALLINT PRIMARY KEY IDENTITY(100,1), --  IDENTITY e o auto incremento, o
primeiro codigo será o 100 e vai pulando de 1 em 1
Nome_Livro VARCHAR(50) NOT NULL, -- NOT NULL. OBRIGATORIO DIGITAR
ISBN VARCHAR(50) NOT NULL UNIQUE, -- O valor do campo ISBN n pode repetir, devido
unique
ID_Autor SMALLINT NOT NULL, --
Data_Pub DATETIME NOT NULL, --
Preco_Livro MONEY NOT NULL --
);

CREATE TABLE tbl_autores(
ID_Autor SMALLINT PRIMARY KEY,
Nome_Autor VARCHAR(50),
SobreNome_Autor VARCHAR(60)
```

```
);

CREATE TABLE tbl_editoras(
ID_Editora SMALLINT PRIMARY KEY IDENTITY,
Nome_Editora VARCHAR(50) NOT NULL
);

--Podemos visualizar informações sobre a tabela criada
--com a ajuda da stored procedure sp_help, a qual informa
-- os índices, chaves e campos, atributos e tipos de atributos
--de uma tabela:

sp_help tbl_Livro;
sp_help tbl_autores;
sp_help tbl_editoras;
sp_help tbl_generos;

--Não é possível alterar uma coluna existente para configurar IDENTITY.
--IDENTITY significa auto incremento permite que um número único seja gerado
--automaticamente quando um novo registro é inserido em uma tabela.

--Excluir a coluna ID_Autor da tabela tbl_livros:
ALTER TABLE tbl_Livro
DROP COLUMN ID_Autor;

/*PARA EXCLUIR UMA CONSTRAINT
ALTER TABLE tabela
DROP CONSTRAINT nome_constraint;*/

SELECT * FROM tbl_Livro;

--Adicionando uma nova coluna a tabela tbl_livros
ALTER TABLE tbl_Livro
ADD coluna_teste SMALLINT NOT NULL;

--Excluindo a coluna coluna_teste da tabela tbl_livros:
ALTER TABLE tbl_Livro
DROP COLUMN coluna_teste;


--Adicionar a coluna ID_Autor à tabela tbl_livros e configurá-la como chave
--estrangeira da coluna ID_Autor da tabela tbl_autores:
--Estou dizendo que a chave estrangeira vai ser chamada "fk_ID_Autor", a coluna
```

**--ID_autor será a chave estrangeira que faz referencia a tabela autores**
**/*Vai na tabela de autores e vai pegar o mesmo campo da chave estrangeira "ID_Autor" e vai fazer a conexão*/**

```
ALTER TABLE tbl_Livro
ADD ID_Autor SMALLINT NOT NULL
CONSTRAINT fk_ID_Autor FOREIGN KEY (ID_Autor) REFERENCES tbl_autores;
```

**/*Adicionar a coluna ID_editora à tabela tbl_livros e configurá-la como chave estrangeira da coluna ID_editora da tabela tbl_editoras:*/**

```
ALTER TABLE tbl_Livro
ADD ID_editora SMALLINT NOT NULL
CONSTRAINT fk_id_editora FOREIGN KEY (ID_editora) REFERENCES tbl_editoras ;
```

**--Alterando o tipo de dados da coluna ID_Autor para DATETIME, antes de ter dados dentro dessa tabela:**

```
ALTER TABLE tbl_Livro
ALTER COLUMN Data_Pub DATETIME;
```

**/*Não e aconselhavel, alterar a estrutura da tabela quando você já tem dados inserido na tabela*/**

**/*Adicionando uma chave primaria em uma tabela já existente que não tem chave primaria**
**A coluna "ID_Cliente deve existir antes de ser transformada em chave primária*/**

```
ALTER TABLE TBL_CLIENTES
ADD CONSTRAINT pk_id_cliente PRIMARY KEY (ID_Cliente);
```

**--Excluir uma Tabela**
**/*Si a tabela estiver conectadas a outras tabelas por meio de restrições (chave primarias,chave estrangeira, relacionamentos, não e possivel excluir as tabelas**
**Primeiro terá que excluir os relacionamentos, para depois excluir a tabela*/**

```
DROP TABLE TBL_CLIENTES;
```

**--Criar uma nova tabela no banco de dados db_Biblioteca, para armazenar gêneros dos livros.**

```
CREATE TABLE tbl_generos (
ID_Genero Tinyint IDENTITY,
Genero VARCHAR(25)
CONSTRAINT pk_id_genero PRIMARY KEY (ID_Genero)
);
```

**--Criar uma nova coluna na tabela de livros para realizar o relacionamento com a tabela de gêneros.**

```
ALTER TABLE tbl_Livro
ADD ID_Genero Tinyint NOT NULL
CONSTRAINT fk_id_genero FOREIGN KEY (ID_Genero) REFERENCES tbl_generos;


INSERT INTO tbl_generos (Genero)
VALUES
('Ficção'),
('Romance'),
('Aventura'),
('Técnico'),
('Suspense');

INSERT INTO tbl_autores (ID_Autor, Nome_Autor)
VALUES (6, 'Daniel Defoe');

INSERT INTO tbl_editoras (Nome_Editora)
VALUES
('Penguin Classics');


SELECT * FROM tbl_Livro;
SELECT * FROM tbl_generos;
SELECT * FROM tbl_autores;
SELECT * FROM tbl_editoras;


DROP TABLE tbl_autores;
DROP TABLE tbl_editoras;
DROP TABLE tbl_generos;
DROP TABLE tbl_Livro;
```

**--1° Maneira de inserir dados nas tabelas**

```
INSERT INTO tbl_Autores (ID_Autor, Nome_Autor, SobreNome_Autor)
VALUES (1, 'Daniel', 'Barret');

INSERT INTO tbl_Autores (ID_Autor, Nome_Autor, SobreNome_Autor)
VALUES  (2, 'Gerald', 'Carter');
```

**--2° Maneira de inserir dados nas tabelas,os dados que vou inserir necessitam está na**

**ordem dos campos das tabelas**
INSERT INTO tbl_Autores VALUES  (3, 'Mark', 'Sobell');

**--3° Maneira de inserir dados nas tabelas, inserir varios dados de uma vez somente**
INSERT INTO tbl_Autores (ID_Autor, Nome_Autor, SobreNome_Autor)
VALUES
(4, 'William', 'Stanek'),
(5, 'Richard', 'Blum');

**-- Não preciso colocar o ID do campo "ID_EDITORA" porque esse campo e auto_increment**
INSERT INTO tbl_Editoras (Nome_Editora)
VALUES
('Prentice Hall'),
('O´Reilly'),
('Microsoft Press'),
('Wiley');

**/*Deixei a tabela de Livros por ultimo, devido essa tabela ter as chaves estrangeiras criadas, tem dois campos/colunas na tabela de livros que dependem de campos/colunas na tabela de Autores e editoras**
**Na tabela de editoras o campo "Id_editora", na tabela de autor o campo "ID_autor" são chaves primarias.**
**Portanto para eu cadastrar um livro em outra tabela eu preciso ter os valores das editoras e dos autores já configurados/inseridos na tabela respectivas, pois caso isso não sejá realizado eu não consigo cadastrar um valor na tabela livros que não existe na tabela autores por conta da chave estrangeira que foi criada**
**Não coloco ID_Livro devido ser auto increment*/**
INSERT INTO tbl_Livro (Nome_Livro, ISBN, Data_Pub, Preco_Livro, ID_Autor, ID_editora, ID_Genero)
VALUES
('Linux Command Line and Shell Scripting','143856969','20091221', 68.35, 5, 4, 4),
('SSH, the Secure Shell','127658789','20091221', 58.30, 1, 2, 4),
('Using Samba','123856789','20001221', 61.45, 2, 2, 4),
('Fedora and Red Hat Linux','123346789', '20101101', 62.24, 3, 1, 4),
('Windows Server 2012 Inside Out','123356789','20040517', 66.80, 4, 3, 4),
('Microsoft Exchange Server 2010','123366789','20001221', 45.30, 4, 3, 4);

**--É importante frisar que devemos primeiro inserir dados nas tabelas que**
**--não possuem campos dependentes de campos de outras tabelas, ou seja,**
**--em tabelas que não possuem chaves estrangeiras**

**--Criando uma tabela de testes chamada "PETSHOP"**

```
CREATE TABLE TBL_PETSHOP(
ID_ANIMAL INT PRIMARY KEY IDENTITY(100,1),
raca_animal VARCHAR (30) NOT NULL,
porte VARCHAR (10) NOT NULL
);

INSERT INTO TBL_PETSHOP (raca_animal, porte)
VALUES
('PitBull', 'Grande'),
('Salsicha', 'Pequeno'),
('Pug', 'Pequeno');

SELECT * FROM TBL_PETSHOP;
```

**--TRUNCATE TABLE Remove todas as linhas/dados/registros de uma tabela**

```
SELECT COUNT(*) AS Numero_de_linhas_Tabela_PETSHOP_Antes FROM TBL_PETSHOP;
GO
TRUNCATE TABLE TBL_PETSHOP
GO
SELECT COUNT(*) AS Numero_de_linhas_Tabela_PETSHOP_Depois FROM TBL_PETSHOP;
```

**/*Verificando quantos registros/linhas/dados eu tenho na tabela livros*/**

```
SELECT COUNT(*) AS Numero_de_linhas_Tabela_de_Livros FROM tbl_Livro;
```

**--Consultando os campos de uma tabela**

```
SELECT Nome_Autor from tbl_Autores;
```

**--Consultando todos os campos/colunas da tabela**

```
SELECT * FROM tbl_Autores;

SELECT Nome_Livro FROM tbl_Livro;

SELECT Nome_Livro, ID_Autor FROM tbl_Livro;

SELECT Nome_Livro, ISBN
FROM tbl_Livro
ORDER BY Nome_Livro;
```

**--ORDER BY - Consultas com ordenação de Colunas**

**--ASC -ORDEM ASCENDENTE**
**--DESC - ORDEM DESCENDENTE (INVERSA)**

**--SELECT colunas from tabela**
**--ORDER BY coluna_a_ordenar**

**--Ordenando pelo campo "Nome_Livro" de forma alfabetica**
SELECT * FROM tbl_Livro
ORDER BY Nome_Livro;

**--Ordenando pelo campo "Nome_Livro" de forma alfabetica**
SELECT * FROM tbl_Livro
ORDER BY Nome_Livro ASC;

**--Ordenando pelo campo "Nome_Livro" de forma alfabetica,mas de forma inversa do "z" para o "a"**
SELECT * FROM tbl_Livro
ORDER BY Nome_Livro DESC;

**--Ordenando pelo campo "Nome_Livro" de forma alfabetica**
SELECT Nome_autor FROM tbl_autores
ORDER BY Nome_autor;

**--Quando utilizamos "ORDER BY" e não colocamos "ASC" ou "DESC", por padrão**
**--será ordenado de forma alfabetica**

**--Ordenando pelo campo "ID_Autor" de forma decrescente, ou sejá, do maior ID para o menor ID**
SELECT Nome_autor, ID_Autor FROM tbl_autores
ORDER BY ID_Autor desc;

**--Ordenando pelo campo "ID_Autor" de forma crescente, ou sejá, do menor ID para o maior ID**
SELECT Nome_autor, ID_Autor FROM tbl_autores
ORDER BY ID_Autor ASC ;

**--Ordenando pelo campo "Data_Pub" de forma decrescente, ou sejá, da maior data de publicação" para a menor**
SELECT Nome_Livro, ID_Livro, Data_Pub FROM tbl_Livro
ORDER BY Data_Pub desc;

**--Ordenando pelo campo "Data_Pub" de forma crescente, ou sejá, da menor "data de**

**publicação" para a maior**
SELECT Nome_Livro, ID_Livro, Data_Pub FROM tbl_Livro
ORDER BY Data_Pub ASC ;

USE db_Biblioteca;

**--Mostra todos os registros de autores que publicaram algum livro**
SELECT ID_Autor FROM tbl_Livro;

**--DISTINCT permite exibir somente valores diferentes, sem ter valores iguais**
**--Mostra apenas valores unicos nas consultas**
SELECT DISTINCT ID_Autor FROM tbl_Livro;

**--WHERE - Filtrando registros/informações em uma consulta**
SELECT * FROM tbl_Livro WHERE ID_Autor = '1';

SELECT ID_Autor, Sobrenome_Autor FROM tbl_autores
WHERE Sobrenome_Autor = 'Stanek';

**-- Operadores AND e OR**
**--AND - Mostra um registro se ambas as condições forem verdadeiras**
**--OR - Mostra um registro se pelo menos uma das condições for verdadeira**

SELECT * FROM tbl_Livro
WHERE ID_Livro > 2 AND ID_Autor < 3;

SELECT * FROM tbl_Livro
WHERE ID_Livro > 107 AND ID_Autor < 3;

SELECT * FROM tbl_Livro
WHERE ID_Livro > 101 OR ID_Autor < 3;

**--UPDATE - permite atualizar dados em uma**
**--coluna de um registro em uma tabela, ou todas as**
**--colunas em todos os registros na tabela.**

**/***
**SINTAXE UPDATE**

UPDATE tabela
SET coluna = valor

```
WHERE 'filtro'
*/

SELECT * FROM tbl_Livro;


--Alterando o nome do livro de ID 106 para Basta Sentir :
UPDATE tbl_Livro
SET Nome_Livro = 'Basta Sentir'
WHERE ID_Livro = 106;

--Alterando o preço do livro Using Samba para R$ 65.43:
UPDATE tbl_Livro
SET Preco_Livro = 65.43
WHERE Nome_Livro = 'Using Samba';

SELECT * FROM tbl_Autores;

--Alterando o sobrenome do autor de id 2 (Gerald Carter) para Carter Jr.
UPDATE tbl_Autores
SET Sobrenome_Autor = 'Carter Jr.'
WHERE ID_Autor =2;

--Alterando o sobrenome do autor de id 6.
UPDATE tbl_Autores
SET Sobrenome_Autor = ' Victor.'
WHERE ID_Autor =6;

SELECT * FROM tbl_Livro;

--Alterar o ISBN para 654738322 e o preço para R$ 71,20, do livro de ID igual a 106:
UPDATE tbl_Livro
 SET Preco_Livro = 71.20,
 ISBN = '654738322'
 WHERE ID_Livro = 106;

--SELECT TOP
-- Usado para especificar o número de registro a retornar
-- Útil para tabelas com muitos registros

--Retornar os nomes dos primeiros 10% de livros encontrados na tabela de livros
USE db_Biblioteca;
GO
```

```
SELECT TOP 10 PERCENT Nome_Livro
FROM tbl_Livro;
```

**--Retornar os nomes do três primeiros livros da tabela de livros**

```
SELECT TOP (3) Nome_Livro
FROM tbl_Livro;
```

**--Retornar os nomes dos primeiros 10% de livros encontrados na tabela de livros (por ordem alfabética), ordenados por nome do livro:**

```
SELECT TOP (10) PERCENT Nome_Livro
FROM tbl_Livro ORDER BY Nome_Livro;
```

**--Retornar os nomes do três primeiros livros da tabela de livros (por ordem alfabética), ordenados por nome do livro:**

```
SELECT TOP (3) Nome_Livro
FROM tbl_Livro ORDER BY Nome_Livro ASC;
```

**--Retornar os nomes do três últimos livros da tabela de livros (por ordem alfabética), ordenados por nome do livro:**

```
SELECT TOP (3) Nome_Livro
FROM tbl_Livro ORDER BY Nome_Livro DESC;
```

**--Retornar os nomes e os IDs dos três primeiros livros cadastrados na tabela:**

```
SELECT TOP (3) Nome_Livro, ID_Livro
FROM tbl_Livro;
```

**--ALIAS**
**--Nomes alternativos para colunas no SQL Server**

**--Retornar os dados da coluna Nome_Livro exibindo como cabeçalho de coluna a palavra Livro, simplesmente:**

```
SELECT Nome_Livro
AS Livro
FROM tbl_Livro;
```

**--Podemos também aplicar um alias a uma coluna sem a necessidade de usar a**
**--palavra AS, usando a sintaxe alternativa apresentada acima. Para isso, basta**
**--inserir o alias desejado logo após o nome da coluna, sem separação por vírgulas.**

```
SELECT Nome_Livro Livro
FROM tbl_Livro;
```

**/*Para aplicar aliases em mais de uma coluna, basta acrescentá-las**

**normalmente, separando-as por vírgulas, e incluindo os alias logo após o nome de cada coluna respectiva.**
**Além disso, podemos criar alias usando palavras compostas, incluindo espaços, bastando para isso envolver o alias entre aspas. O exemplo a seguir mostra as duas possibilidades juntas:*/**

```
SELECT Nome_Livro Livro, Preco_Livro 'Preço do Livro'
FROM tbl_Livro;
```

**--OPERADOR UNION**

**--Unir resultados de declarações SELECT**
**/*Permite combinar duas ou mais declarações SELECT**
**Cada declaração SELECT deve ter o mesmo número de colunas, tipo de dados e ordem das colunas*/**
**/*SINTAXE**

```
SELECT colunas FROM tabela1
UNION
SELECT colunas  FROM tabela2*/
```

```
SELECT ID_Autor FROM tbl_autores
UNION
SELECT ID_Autor FROM tbl_Livro;
```

**/*Vai aparecer os ID_Autor  da tabela tbl_autores**
**mais os ID_Autor  da tabela tbl_Livro. Nesse caso vai aparecer ID_Autor repetidos */**

```
SELECT ID_Autor FROM tbl_autores
UNION ALL
SELECT ID_Autor FROM tbl_Livro;
```

```
SELECT Nome_Autor FROM tbl_autores
UNION
SELECT Nome_Livro FROM tbl_Livro;
```

**--SELECT INTO**
**--Criar uma nova tabela a partir de uma tabela existente**
**--Seleciona dados de uma ou mais tabelas e os insere em uma tabela diferente**
**--Pode ser usada para criar cópias de backup de tabelas, já com seus dados incluidos**
**/*SINTAXE**

```
SELECT * INTO nova_tabela
FROM tabela_atual
```

```
*/

--Estou criando uma nova tabela chamada "LivroAutor" que terão as colunas
--"Nome_Livro" e "Id_Autor", onde os dados dessa nova
--tabela viram da tabela "tbl_Livro", mas eu não vou colocar todos os livros nessa
--tabela nova, só os livros cujo o "ID_Livro" sejá maior que 2
SELECT Nome_Livro, Id_Autor
INTO LivroAutor
FROM tbl_Livro
WHERE ID_Livro >2;

SELECT ID_Livro, Nome_Livro, ISBN
INTO Livro_ISBN
FROM tbl_Livro
WHERE ID_Livro >2;

--Fazendo um Backup da tabela tbl_Livro, vai fazer backup dela completa,
--vai gravar todos dados/registro da tbl_Livro na nova tabela "tbl_Livro_backup
SELECT *
INTO tbl_Livro_backup
FROM tbl_Livro;

SELECT * FROM Livro_ISBN;
SELECT * FROM LivroAutor;
SELECT * FROM tbl_Livro_backup;
SELECT * FROM  tbl_Livro;

DROP TABLE Livro_ISBN;

--FUNÇÕES AGREGADAS -SUM, COUNT, MAX, MIN, AVG

--MIN = Valor Mínimo
--MAX = Valor Máximo
--AVG = Média Aritmética
--SUM = Total(Soma)
--COUNT = Contar quantidade de itens

--Quantos registros/linhas eu tenho nessa tabela
SELECT COUNT(*) FROM tbl_autores;

--Conta, quantos números de nomes de autores tem na tabela de "autores"
--Pode haver mudanças na tabela, pois as vezes o campo pode aceitar "Null"
--ou seja, um campo que permite que vc não digite uma informação quando
```

**--você cadastra uma informação**
SELECT COUNT(Nome_Autor)  FROM tbl_autores;

**--Vai trazer o valor máximo da coluna "Preco_Livro" da tabela "tbl_Livro"**
SELECT MAX(Preco_Livro) AS PreçoMaximo FROM tbl_Livro;

**--Vai trazer o valor minimo da coluna "Preco_Livro" da tabela "tbl_Livro", ou sejá livro mais barato**
SELECT MIN(Preco_Livro) AS PreçoMinimo FROM tbl_Livro;

**--Nesse caso, quero saber o valor medio/preco medio de todos os livros que eu tenho**
**--AVG vai somar todos os valores que vai encontrar na coluna "Preco_Livro"**
**-- e vai dividir pelo número de itens/linhas**
SELECT AVG(Preco_Livro) AS PreçoMedio FROM tbl_Livro;

**--Somando todos os precos do livro da tabela de livros**
SELECT SUM(Preco_Livro) AS PrecoTotal FROM tbl_Livro;

**--BETWEEN - SELEÇÃO DE INTERVALOS**
**--Intervalo de Filtragem**

**/*SINTAXE**

SELECT colunas FROM tabela
WHERE coluna BETWEEN valor1 AND valor2
*/

**--Selecionar todos os campos da "tbl_Livro",mas os registros onde as datas**
**--de publicações esteja entre "20040517" e "20100517"**
SELECT * FROM tbl_Livro
WHERE Data_Pub BETWEEN '20040517' AND '20100517';

SELECT Nome_Livro AS Livro, Preco_Livro AS Preço FROM tbl_Livro
WHERE Preco_Livro BETWEEN '40.00' AND '60.00';

**--LIKE e NOT LIKE - Filtragem por padrões especificos**

USE db_Biblioteca;

**--Quero descobrir quais livros começam com a letra S**
SELECT Nome_livro
FROM tbl_Livro

```
WHERE Nome_Livro LIKE 'S%';

--Quero descobrir quais livros terminam com a letra G
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '%g';

--Quero descobrir livros cuja a segunda letra do nome, e a letra "i", não importa
-- a primeira letra, só importa a segunda letra
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '_i%';

--Pesquisando o nome de livros que a primeira letra e S ou L, não importa o que vem
depois
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '[SL]%';

--Pesquisando o nome de livros onde as ultimas letras seja "g" ou "l"
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '%[gl]';

--Pesquisando o nome de livros onde a segunda letra seja "i" ou "S" e não importa o
que vem depois
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '_[iS]%';

--Pesquisando livros que a segunda letra seja "i" e a quinta letra seja "o"
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro LIKE '_i__o%';

--Pesquisando livros que não comecem com a letra "m"
SELECT Nome_livro
FROM tbl_Livro
WHERE Nome_Livro NOT LIKE 'M%';

--JOINS
--É usada para obter dados provenientes de duas ou mais tabelas
--baseado em um relacionamento entre colunas nestas tabelas
```

**--INNER JOIN - Retorna linhas quando houver pelo menos uma correspondência**
**--em ambas as tabelas**

**--OUTER JOIN - Retorna linhas mesmo quando não houver pelo menos uma**
**--correspondência em uma das tabelas (ou ambas)**

**--O OUTER JOIN divide-se em LEFT JOIN, RIGHT JOIN e FULL JOIN**

/*
**SINTAXE**

SELECT colunas
FROM tabela1
INNER JOIN tabela2
ON tabela1.coluna = tabela2.coluna

*/

SELECT * FROM tbl_Livro;

SELECT * FROM tbl_autores;

**--Juntando as tabelas livros e a tabela autores**
SELECT * FROM tbl_Livro
INNER JOIN tbl_autores
ON tbl_Livro.ID_Autor = tbl_autores.ID_Autor; --Relacionamento entre as tabelas e a chave primaria e a chave estrangeira

SELECT tbl_Livro.Nome_Livro, tbl_Livro.ISBN, tbl_autores.Nome_Autor
FROM tbl_Livro
INNER JOIN tbl_autores
ON tbl_Livro.ID_Autor = tbl_autores.ID_Autor;

**--Utilizando Aliases**
SELECT L.Nome_Livro, E.Nome_editora
FROM tbl_Livro AS L
INNER JOIN tbl_editoras AS E
ON L.ID_editora = E.ID_editora;

**--LEFT JOIN - Retorna todas as linhas da tabela à esquerda, mesmo se não**
**-- houver nenhuma correspondência na tabela à direita**

**/*SINTAXE**

```
SELECT coluna
FROM tabela_esquerda
LEFT JOIN tabela_direita
ON tabela_esquerda.coluna = tabela_direita.coluna
*/
SELECT * FROM tbl_autores;

SELECT * FROM tbl_Livro;

SELECT * FROM tbl_autores
LEFT JOIN tbl_Livro
ON tbl_Livro.ID_Autor = tbl_autores.ID_Autor;

SELECT * FROM tbl_Livro --Tabela esquerda
LEFT JOIN tbl_autores
ON tbl_Livro.ID_Autor = tbl_autores.ID_Autor;
```

**--RIGHT JOIN - Retorna todas as linhas da tabela à direita, mesmo se não**
**--houver nenhuma correspondência na tabela à esquerda**

**--FULL JOIN - Retorna linhas quando houver uma correspondência em qualquer**
**-- uma das tabelas. É uma combinação de LEFT e RIGHT JOINS**

```
INSERT INTO tbl_Livro( Nome_Livro, ISBN, Data_Pub, Preco_Livro,ID_editora, ID_Genero )
VALUES
('O fats dos milionarios','133344786' ,' 2023-10-02 00:00:00.000', 23.55,  3, 3),
('O full dos pobres','111344886' , '2020-10-02 00:00:00.000', 44.4 ,3, 3);
```

**-- Arquivo teste.sql**

```
CREATE DATABASE DB_TESTE;

DROP TABLE TBL_CLIENTES;

CREATE TABLE TBL_CLIENTES(
ID INT PRIMARY KEY IDENTITY(1,1),
NOME VARCHAR(50),
email VARCHAR(50),
data_nascimento DATE
```

```
);

USE DB_TESTE;

INSERT INTO TBL_CLIENTES ( NOME, email, data_nascimento)
VALUES
('João Fernando', '         joao@gmail.com','2000-01-01'),
('Maria Carvalho','maria@gmail.com','2001-10-01'),
('Bruna dos Santos','bruna@gmail.com','2001-10-01'),
('Pedro da Silva','pedro@gmail.com','2001-11-01'),
('Tiago','tiago@gmail.com','2001-10-02'),
('Bruno Henrique', 'bruno@gmail.com','2000-01-01'),
('Pedro Pereira','pedro@gmail.com','2001-10-01'),
('Fernando Luiz','fernando@gmail.com','1999-02-03'),
('Luana Pereira','luana@gmail.com','2005-01-30'),
('Bruno Henrique', 'bruno@gmail.com','1970-07-09'),
('Alan Dias','alan@gmail.com','1963-04-01'),
('Michele Braga','michele@gmail.com','1999-11-01');

SELECT * FROM TBL_CLIENTES;

/*Suponhamos que seja necessário trazer na consulta todos os clientes
ordenados pelo nome.
As informações foram ordenadas de forma ascendente/ALFABETICAS, ou seja,
ao não informar o parâmetro de ordenação, por padrão o valor ASC é acionado*/
SELECT * FROM TBL_CLIENTES
ORDER BY NOME;

/*Suponhamos que seja necessário trazer na consulta todos os clientes
ordenados pelo nome.
As informações foram ordenadas de forma descendente/ALFABETICAS de forma
inversa, ou seja,
vai primeiro mostrar os nome com Z e por ultimo os que começam com A */
SELECT * FROM TBL_CLIENTES
ORDER BY NOME DESC;

/*Neste exemplo o resultado será ordenado de forma ascendente,ou sejá,
do menor ID para o Maior*/
SELECT * FROM TBL_CLIENTES
ORDER BY ID ASC;

/*Neste exemplo o resultado será ordenado de forma descendente,ou sejá,
do maior ID para o Menor*/
```

```
SELECT * FROM TBL_CLIENTES
ORDER BY ID DESC;
```

**--Consulta onde o resultado será ordenado tanto pelo nome quanto pela data de nascimento.**
**/*Consulta onde iremos primeiramente ordenar as consultas pelo nome em seguida pela data de  nascimento*/**

```
SELECT NOME, data_nascimento
FROM TBL_CLIENTES
ORDER BY NOME, data_nascimento ASC;

SELECT NOME, data_nascimento
FROM TBL_CLIENTES
ORDER BY NOME ASC;

SELECT NOME, data_nascimento
FROM TBL_CLIENTES
ORDER BY data_nascimento ASC;
```

## MySQL – Constraints (Restrições) Primary Key, FK, Default, etc – 06

**SQL Constraints (Restrições) no MySQL**

- As Restrições são regras aplicadas nas colunas de uma tabela.
- São usadas para limitar os tipos de dados que são inseridos.
- Podem ser especificadas no momento de criação da tabela (CREATE) ou após a tabela ter sido criada (ALTER)

As principais constraints são as seguintes:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- DEFAULT

**NOT NULL**

- A constraint NOT NULL impõe a uma coluna a NÃO aceitar valores NULL.

- Ou seja, a constraint NOT NULL obriga um campo a sempre possuir um valor.
- Deste modo, não é possível inserir um registro (ou atualizar) sem entrar com um valor neste campo.

**UNIQUE**

- A restrição UNIQUE identifica de forma única cada registro em uma tabela de um banco de dados.
- As constraints UNIQUE e PRIMARY KEY garantem a unicidade em uma coluna ou conjunto de colunas.
- Uma constraint PRIMARY KEY automaticamente possui uma restrição UNIQUE definida, portanto não é necessário especificar essa constraint neste caso.

- É possível termos várias constraints UNIQUE em uma mesma tabela, mas apenas uma Chave Primária por tabela (lembrando que uma PK pode ser composta, ou seja, constituída por mais de uma coluna – mas ainda assim, será uma única chave primária).

**PRIMARY KEY**

- A restrição PRIMARY KEY (Chave Primária) identifica de forma única cada registro em uma tabela de banco de dados.
- As Chaves Primárias devem sempre conter valores únicos.
- Uma coluna de chave primária não pode conter valores NULL
- Cada tabela deve ter uma chave primária e apenas uma chave primária.

**FOREIGN KEY**

Uma FOREIGN KEY (Chave Estrangeira) em uma tabela é um campo que aponta para uma chave primária em outra tabela. Desta forma, é usada para criar os relacionamentos entre as tabelas no banco de dados.

**Veja um exemplo de restrição Foreign Key aplicada:**

**CONSTRAINT fk_ID_Autor FOREIGN KEY (ID_Autor)**

**REFERENCES tbl_autores(ID_Autor)**

Neste exemplo a chave primária está na tabela tbl_autores e uma chave estrangeira de nome ID_Autor foi criada na tabela atual, usando o nome fk_ID_Autor

**DEFAULT**

- A restrição DEFAULT é usada para inserir um valor padrão especificado em uma coluna.

O valor padrão será adicionado a todos os novos registros caso nenhum outro valor seja especificado na hora de inserir dados.

**--OUTRO ARQUIVO SQL**

```
/*Criando um banco de dados*/
CREATE DATABASE db_Biblioteca;

/*Verificando os bancos de dados existentes*/
SHOW DATABASES;

/*O comando USE instrui o SGBDR a utilizar o banco de dados especificado para rodar os comandos.*/
USE db_Biblioteca;

/*Comando para visualizar o banco de dados selecionado no momento*/
SELECT DATABASE();

/*Exluindo o banco de dados*/
DROP DATABASE db_Biblioteca;

/*Criando tabelas no Banco de Dados para os livros*/
CREATE TABLE IF NOT EXISTS tbl_livro (/*Crie essa tabela caso ela não exista*/
ID_Livro SMALLINT AUTO_INCREMENT PRIMARY KEY,
nome_Livro VARCHAR(70) NOT NULL,/*Not Null e uma coluna que não aceitas valores nulos,o campo deve possuir um valor*/
ISBN13 CHAR(13),
ISBN10 CHAR(10),
ID_Categoria SMALLINT,
ID_Autor SMALLINT NOT NULL,
Data_Pub DATE NOT NULL,
Preco_Livro DECIMAL(6,2) NOT NULL
);

/*Criando tabelas no Banco de Dados para armazenar os dados dos Autores*/
CREATE TABLE tbl_autores (
ID_Autor SMALLINT PRIMARY KEY,
Nome_Autor VARCHAR(50) NOT NULL,
sobrenome_Autor VARCHAR(60) NOT NULL
);
```

**/*Criando tabelas no Banco de Dados para armazenar as categorias de livros*/**

```
CREATE TABLE tbl_categorias (
ID_Categoria SMALLINT PRIMARY KEY,
Categoria VARCHAR(30) NOT NULL
);
```

**/*Criando tabela no Banco de Dados para armazenar os dados das editoras*/**

```
CREATE TABLE tbl_editoras (
ID_Editora SMALLINT PRIMARY KEY AUTO_INCREMENT,
Nome_Editora VARCHAR(50) NOT NULL
);
```

**/*O auto incremento permite que um número único seja gerado quando um novo registro é inserido em uma tabela.*/**

**/*Tabela de Teste para Estudar a utilização do "auto incremento"*/**

```
CREATE TABLE tbl_teste_incremento (
Codigo SMALLINT PRIMARY KEY AUTO_INCREMENT,
Nome VARCHAR(20) NOT NULL
) AUTO_INCREMENT = 15;/*Quero que o valor da coluna "Codigo" inicie em 15*/

INSERT INTO tbl_teste_incremento (Nome) VALUES ('Ana');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Maria');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Julia');
INSERT INTO tbl_teste_incremento (Nome) VALUES ('Joana');

SELECT * FROM tbl_teste_incremento;
```

**/*verificar o valor de auto_incremento mais atual armazenado na tabela tbl_teste_incremento no banco de dados*/**

```
SELECT MAX(Codigo)
FROM tbl_teste_incremento;
```

**/*Alterando o valor do auto_incremento do proximo registro a ser armazendo na tabela para o valor 90*/**

```
ALTER TABLE tbl_teste_incremento
AUTO_INCREMENT = 90;

INSERT INTO tbl_teste_incremento (Nome) VALUES ('Bruno');
```

**/*Atributo UNSIGNED na criação da tabela impede que valores negativos sejam armazenados na coluna**

**Atributo ZEROFILL faz com que o número seja exibido com preenchimento de zeros Ã  esquerda em uma consulta*/**

**/*Excluindo a coluna ID_autor da tabela tbl_livro;*/**

```
ALTER TABLE tbl_livro
DROP COLUMN ID_autor;

SELECT * FROM tbl_livro;
```