SQL SERVER

1. **SELECT – Comando para efetuar busca de dados**

Ex:

```
SELECT coluna1, coluna
FROM tabela;
```

2. **DISTINCT – Comando para omitir os dados duplicados**

Ex:

```
SELECT DISTINCT coluna1, coluna2
FROM tabela;
```

3. **WHERE - Comando para extração de dados (filtro)**

Ex:

```
SELECT coluna1, coluna2
FROM tabela
WHERE condição;
```

OPERADORES PARA USO DO WHERE -	DESCRIÇAO
=	IGUAL
>	MAIOR QUE
<	MENOR QUE
>=	MAIOR QUE OU IGUAL
<=	MENOR QUE OU IGUAL
<>	DIFERENTE DE
AND	OPERADOR LÓGICO E
OR	OPERACOR LÓGICO OU

4. **COUNT – Comando usado para retornar o número de linhas de determinada condição**

Ex:

SELECT COUNT (coluna)

FROM tabela;

5. **TOP – Comando para limitar a quantidade de dados de um SELECT**

EX:

SELECT TOP 10 coluna

FROM tabela

6. **ORDER BY – Comando para ordenar colunas em ordem crescente ou decrescente**

(asc / desc)

Ex:

SELECT coluna1, coluna2

FROM tabela

ORDER BY coluna1 asc/desc

7. **BETWEEN – Comando que é usado para encontrar o valor entre um valor mínimo e valor máximo.**

Valor BETWEEN mínimo AND máximo

Ex:

SELECT coluna

FROM tabela

WHERE coluna BETWEEN valor1 AND valor2;

8. **IN – Usado o operador IN junto com o WHERE, para verificar se um valor corresponde com qualquer valor passado na lista de valores.**

Ex:

SELECT coluna

FROM tabela

WHERE coluna IN (valor1, valor2, valor3);

9. LIKE – Operador LIKE

O operador LIKE é utilizado para buscar por uma determinada string dentro de um campo com valores textuais. Com ele podemos, por exemplo, buscar os registros cujo NOME inicia com uma determinada palavra, ou contém um certo texto.

Para efetuar esse tipo de consulta, podemos utilizar também o caractere % para indicar um "coringa", ou seja, um texto qualquer que pode aparecer no campo. Sua sintaxe padrão é a seguinte:

SELECT colunas FROM tabela WHERE campo LIKE 'valor'

Nessa instrução, o "valor" pode ser informado de várias formas:

texto: Nesse caso, serão retornados todos os registros que contêm no campo buscado exatamente o "texto" informado no filtro. O funcionamento aqui é equivalente a utilizar o operador de igualdade (=);

%texto%: Serão retornados os registros que contêm no campo buscado o "texto" informado. Por exemplo, podemos buscar os nomes que contêm "Santos", ou que contêm uma sílaba ou letra específica. O registro com nome "Luis da Silva", por exemplo, contém o termo "da", então atenderia ao filtro '%da%';

%texto: Serão retornados os registros cujo valor do campo filtrado termina com o "texto" informado. O %, nesse caso, indica que pode haver qualquer valor no começo do campo, desde que ele termine com o "texto". Por exemplo, o registro com nome "Luis da Silva" atenderia ao filtro '%Silva';

texto%: Serão retornados os registros cujo valor do campo filtrado começa com o "texto" informado. Dessa vez, o % indica que após o "texto" pode haver qualquer valor. Por exemplo, o registro com nome "Luis da Silva", atenderia ao filtro 'Luis%'.

Ex:

SELECT coluna

FROM tabela

WHERE coluna LIKE ‘Eli%’ / ‘%Eli’ / ‘%Eli%’

MIN / MAX / SUM / AVG – Funções de agregação basicamente agregam ou combinam dados de uma tabela em 1 resultado só

Ex:

```
--Traz a soma de valores da coluna determinada
SELECT TOP 10 SUM(LineTotal) AS "Soma"
FROM Sales.SalesOrderDetail
```

```sql
--Traz o Minimo de valor da coluna determinada
SELECT TOP 10 MIN(LineTotal) AS "Minimo"
FROM Sales.SalesOrderDetail

--Traz o maximo de valor da coluna determinada
SELECT TOP 10 MAX(LineTotal) AS "Maximo"
FROM Sales.SalesOrderDetail

--Traz o valor médio do valor da coluna determinada
SELECT TOP 10 AVG(LineTotal) AS "Media"
FROM Sales.SalesOrderDetail
```

10. **GROUP BY – Comando para basicamente dividir o resultado da sua pesquisa em grupos**

Ex:

```sql
SELECT coluna1, funcaoAgregacao(coluna2)

FROM nomeTabela

GROUP BY coluna1;
```

```sql
--Valor da Soma
SELECT SpecialOfferID, SUM(UnitPrice) AS "SOMA"
FROM Sales.SalesOrderDetail
GROUP BY SpecialOfferID

--Quantidade de vendas
SELECT ProductId, COUNT(ProductID) AS "QuantidadeVendidos"
FROM Sales.SalesOrderDetail
GROUP BY ProductID

-- Contagem de vezes que o mesmo nome aparece na tabela
SELECT FirstName, COUNT(FirstName) AS "CONTAGEM"
FROM Person.Person
GROUP BY FirstName

--Para saber a média de preço dos produtos que são pratas(silver)
SELECT color, AVG(ListPrice) AS "MediaPreco"
FROM Production.Product
WHERE Color = 'Silver'
GROUP BY Color

SELECT ProductID,COUNT(ProductId) AS 'Contagem', AVG(OrderQty) AS
'QuantidadeMedia'
FROM Production.WorkOrder
GROUP BY ProductID
```

11. **HAVING – O Having é basicamente muito usado em junção com o group by para filtrar resultados de um agrupamento.**

Ex:

```sql
SELECT coluna1, funcaoAgregacao(coluna2)
```

```
FROM nomeTabela

GROUP BY coluna1
HAVING condição;

--Para saber a contagem de nomes que aparecem maior que 10x

SELECT FirstName, COUNT(FirstName) AS 'Quantidade'
FROM Person.person
GROUP BY FirstName
HAVING COUNT(FirstName) > 10


--Para saber a Soma de valor total de venda com a condição BETWEEN 162000 AND
500000
SELECT ProductId, SUM(Linetotal) AS 'TotalVenda'
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) BETWEEN 162000 AND 500000


AS – Comando usado para dar apelidos as colunas ou tabelas trazendo as
informações

Ex:

SELECT TOP 10 ListPrice AS 'Preço do Produto'
FROM Production.Product


SELECT TOP 10 AVG(ListPrice) AS "Preço Médio"
FROM Production.Product


SELECT FirstName AS 'Nome', LastName AS ' Sobrenome'
FROM Person.Person
```

12. **TIPOS DE JOIN – Comando usado para juntar informações de mais de uma tabela**

12.1. INNER JOIN – Busca informações de duas tabelas trazendo o mesmo campo que está ligado entre elas o campo em comum entre elas (FK -> PK), EX: Utilizando o INNER JOIN você terá como resultado de sua consulta somente os pares de cliente/compra, ou seja, os casos em que uma compra não está ligada a nenhum cliente, ou cliente que não possuem compras, não serão apresentados no resultado.

Ex:

```
-- BusinessEntityId, FirstName, LastName, EmailAddress
```

```
SELECT p.BusinessEntityID, p.FirstName, p.LastName, pe.EmailAddress
FROM Person.Person AS P
INNER JOIN Person.EmailAddress AS PE ON p.BusinessEntityID = pe.BusinessEntityID
```

Ex:

```
--Vamos dizer que nós queremos os nomes dos produtos e as informações de suas
subcategorias
--ListPrice, Nome do Produto, Nome da subcategoria

SELECT ListPrice, pr.Name, pc.Name
FROM Production.Product AS PR
INNER JOIN Production.ProductSubcategory AS PC ON pr.ProductSubcategoryID =
pc.ProductSubcategoryID
```

Ex:

```
--Para fazer a junção de tabelas trazer todas colunas de duas ou mais tabelas
SELECT TOP 10 *
FROM Person.BusinessEntityAddress AS BA
INNER JOIN Person.Address PA ON PA.AddressID = BA.AddressID
```

12.2. LEFT OUTER JOIN ou LEFT JOIN – O LEFT JOIN traz todos os resultados da tabela mais à esquerda e o agrega ao seu valor correspondente das outras tabelas, caso existam. EX: Ou seja, nesse caso a consulta retornaria todos os clientes e suas compras, caso existam. Caso não existam compras para essa cliente os campos relativos à tabela de compras ficariam em branco.

Ex:

```
-- Traz as informações do cartão e crédito da tabela B mesmo que não tenha cartão
de crédito registrado

SELECT *
FROM person.Person AS PP
LEFT JOIN Sales.PersonCreditCard AS PC ON PP.BusinessEntityID =
PC.BusinessEntityID
```

12.3. SELF JOIN – Usado para agrupar dados usando a mesma tabela, para filtrar as informações na mesma tabela.

Ex:

```
--SELECT NOME_COLUNA
--FROM TABELA AS A, TABELA AS B
--WHERE CONDICAO
```

```sql
--Extrair todos os clientes que moram na mesma região
SELECT A.ContactName, A.Region, B.ContactName, B.Region
FROM Customers AS A, Customers AS B
WHERE A.Region = B.Region

--Encontrar (nome e data de contratação) de todos os funcionários que foram
contratados no mesmo ano
SELECT A.FirstName, A.HireDate, B.FirstName, B.HireDate
FROM Employees AS A, Employees AS B
WHERE DATEPART(YEAR, A.HireDate) = DATEPART(YEAR, B.HireDate)

--Extrair da tabela detalhe do pedido [Order Details], quais produtos tem o mesmo
percentual de desconto
SELECT A.ProductID, A.Discount, B.ProductID, B.Discount
FROM [Order Details] AS A, [Order Details] AS B
WHERE A.Discount = B.Discount
```

12.4. UNION – Combina dois ou mais resultados de um select em um resultado apenas.

Ex:

```sql
SELECT coluna1, coluna2

FROM tabela1

UNION

SELECT coluna1, coluna2

FROM tabela2
```

Ex:

```sql
SELECT [ProductID], [Name], [ProductNumber]
FROM Production.Product
WHERE Name like '%Chain%'
UNION
SELECT [productID], [Name], [ProductNumber]
FROM   Production.Product
WHERE Name like '%Decal%'
```

13. DATEPART – Função para extração de dados de data (DATEPART (datepart , date)) – Oque seria DATEPART (dadosExtrair, coluna)

ARGUMENTOS	
datepart	**Abreviações**
year	yy, yyyy
quarter	qq, q

month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw
hour	hh
minute	mi, n
second	ss, s
millisecond	ms
microsecond	mcs
nanosecond	ns
tzoffset	tz
iso_week	isowk, isoww

RETORNO

datepart	**Valor retornado**
year, yyyy, yy	2007
quarter, qq, q	4
month, mm, m	10
dayofyear, dy, y	303
day, dd, d	30
week, wk, ww	44
weekday, dw	3
hour, hh	12
minute, n	15
second, ss, s	32
millisecond, ms	123
microsecond, mcs	123456
nanosecond, ns	123456700
tzoffset, tz	310
iso_week, isowk, isoww	44

Ex:

```
SELECT SalesOrderID, DATEPART(year, OrderDate) AS 'Mês'
```

```
FROM Sales.SalesOrderHeader
```

Ex:

```
SELECT AVG(TotalDue) AS 'MEDIA', DATEPART(month, OrderDate) AS 'MÊS'
FROM Sales.SalesOrderHeader
GROUP BY DATEPART(month, OrderDate)
ORDER BY 'MÊS'
```

14. **Operações em STRING – Usado para manipulação de dados em formato string**

ASCII	NCHAR	STRING_AGG
CHAR	PATINDEX	STRING_ESCAPE
CHARINDEX	QUOTENAME	STRING_SPLIT
CONCAT	REPLACE	STUFF
CONCAT_WS	REPLICATE	SUBSTRING
DIFFERENCE	REVERSE	TRANSLATE
FORMAT	RIGHT	TRIM
LEFT	RTRIM	UNICODE
LEN	SOUNDEX	UPPER
LOWER	SPACE	
LTRIM	STR	

Ex:

```
--Usado para concatenar duas colunas em uma só
SELECT CONCAT(FirstName,' ', LastName)
FROM Person.Person

--Deixa todos  os valores da coluna consultada em maiusculo
SELECT UPPER (FirstName)
FROM Person.Person

--Deixa todos  os valores da coluna consultada em minusculo
SELECT LOWER (FirstName)
FROM Person.Person
```

```sql
--Usado para fazer a contagem de caracteres que a string contém nas células da
coluna informada
SELECT FirstName, LEN(FirstName)
FROM Person.Person

--Comando usado para extração de caracteres da coluna informada, retornando os
devidos caracteres das células (SUBSTRING(coluna, indice, QuantidadeExtração))
SELECT FirstName, SUBSTRING(FirstName, 1, 3)
FROM Person.Person

--Comnando para subtituir valores de uma coluna nas células referidas
(REPLACE(coluna, ValorSubstituir, ValorResultado))
SELECT REPLACE(ProductNumber, '-', '#')
FROM Production.Product
```

15. Funções Matemáticas

Ex:

```sql
--Pode ser feito as operações também com (+, -, /, *) - SUM, AVG, MIN, MAX
SELECT *
FROM sales.SalesOrderDetail

--Usado para arredondar valores ROUND(coluna, casasDecimais)
SELECT ROUND(LineTotal, 2), LineTotal
FROM SALES.SalesOrderDetail

--Usado para resultado da raiz quadrada
SELECT SQRT(LineTotal)
FROM Sales.SalesOrderDetail
```

16. SUBQUERY (SUBSELECT) – Comando usado para fazer Query dentro de outro

Ex:

```sql
-- Monte um relatório de todos os produtos cadastrados que tem preço de venda
acima da média
SELECT *
FROM Production.Product
WHERE ListPrice > (SELECT AVG(ListPrice) FROM Production.Product)
```

Ex:

```sql
--Para saber o nomes dos funcionários que tem o cargo de 'Desing Enginner'
SELECT FirstName
FROM Person.Person
WHERE BusinessEntityID IN (
SELECT BusinessEntityID
FROM HumanResources.Employee
WHERE JobTitle = 'Network Manager')
```

Ex:

```sql
SELECT *
FROM Person.Address
WHERE StateProvinceID IN (
SELECT StateProvinceID
FROM Person.StateProvince
```

```
WHERE Name = 'Alberta')
```

17. Tipos de dados

Tipo de Dados: String

Tipo de Dados	Descrição	Tamanho Máximo	Tamanho (bytes)
char(n)	Tamanho fixo, completado com espaços em bracos	8,000 caracteres	Tamanho Definido
varchar(n)	Tamanho variável com limite	8,000 caracteres	2 bytes + número de caracteres
varchar(max)	Tamanho variável com limite	1,073,741,824 caracteres	2 bytes + número de caracteres
text	Tamanho variável	2GB de dados (texto)	4 bytes + número de caracteres
nchar	Tamanho fixo com espaços em bracos	4,000 caracteres	Tamanho definido x 2
nvarchar	Tamanho variável	4,000 caracteres	
nvarchar(max)	Tamanho variável	536,870,912 caracteres	
ntext	Tamanho variável	2GB de texto	
binary(n)	Tamanho fixo (binário)	8,000 bytes	
varbinary	Tamanho variável (binário)	8,000 bytes	
varbinary(max)	Tamanho variável (binário)	2GB	
image	Tamanho variável (binário)	2GB	

Tipo de Dados: Númericos

Tipo de Dado	Descrição	Tamanho (bytes)
bit	Número Inteiro que pode ser 0, 1 ou NULL	
tinyint	Permite números inteiros de 0 a 255	1 byte
smallint	Permite números inteiros entre -32,768 e 32,767	2 bytes
int	Permite números inteiros entre -2,147,483,648 e 2,147,483,647	4 bytes

bigint	Permite números inteiros entre -9,223,372,036,854,775,808 e 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Precisão de número flutuante e número de escala. Permite número de -10^38 +1 a 10^38 –1. O parâmetro p indica o número total máximo de dígitos que podem ser armazenados (ambos à esquerda e à direita do ponto decimal). p deve ser um valor de 1 a 38. O padrão é 18. O parâmetro s indica o número máximo de dígitos armazenados à direita do ponto decimal. s deve ser um valor de 0 a p. O valor padrão é 0.	5-17 bytes
numeric(p,s)	Precisão de número flutuante e número de escala. Permite número de -10^38 +1 a 10^38 –1. O parâmetro p indica o número total máximo de dígitos que podem ser armazenados (ambos à esquerda e à direita do ponto decimal). p deve ser um valor de 1 a 38. O padrão é 18.	5-17 bytes

	O parâmetro s indica o número máximo de dígitos armazenados à direita do ponto decimal. s deve ser um valor de 0 a p. O valor padrão é 0	
smallmoney	Tipo de "Moeda" de -214,748.3648 a 214,748.3647	4 bytes
money	Tipo de "Moeda" de -922,337,203,685,477.5808 a 922,337,203,685,477.5807	8 bytes
	Precisão de número flutuante de -1.79E + 308 a 1.79E + 308.	
float(n)	O parâmetro n indica se o campo deve conter 4 ou 8 bytes. float (24) contém um campo de 4 bytes e o float(53) mantém um campo de 8 bytes. O valor padrão de n é 53.	4 ou 8 bytes
real	Precisão de número flutuante de -3,40E + 38 a 3,40E + 38	4 bytes

Tipo de Dados: Data

Tipo de Dado	Descrição	Tamanho (bytes)
datetime	De 1 de janeiro de 1753 a 31 de dezembro de 9999 com uma precisão de 3,33 milisegundos	8 bytes
datetime2	De 1º de janeiro de 0001 a 31 de dezembro de 9999 com precisão de 100 nanossegundos	6-8 bytes
smalldatetime	De 1 de janeiro de 1900 a 6 de junho de 2079 com precisão de 1 minuto	4 bytes

date	Armazena apenas uma data. De 1 de janeiro de 0001 a 31 de dezembro de 9999	3 bytes
time	Armazena um tempo apenas para uma precisão de 100 nanosegundos	3-5 bytes
datetimeoffset	O mesmo que datetime2 com a adição de um deslocamento de fuso horário	8-10 bytes
timestamp	Armazena um número único que é atualizado sempre que uma linha é criada ou modificada. O valor do timestamp é baseado em um relógio interno e não corresponde ao tempo real. Cada tabela pode ter apenas uma variável timestamp	

Outros:

Tipo de Dado	Descrição
sql_variant	Armazena até 8.000 bytes de dados de vários tipos de dados, exceto text, ntext e timestamp
uniqueidentifier	Armazena um identificador globalmente exclusivo (GUID)
xml	Armazena dados formatados em XML. Máximo de 2GB
cursor	Armazena uma referência a um cursor usado para operações de banco de dados

table	Armazena um conjunto de resultados para processamento posterior

18. CREATE TABLE

Ex:

CREATE TABLE nomeTabela(

Coluna1 tipo restricaoDaColuna,

Coluna1 tipo restricaoDaColuna,

Coluna1 tipo,

.....

);

Tipos comuns de restrições são:

NOT NULL - cada valor em uma coluna não deve ser NULL

UNIQUE - valor (es) na (s) coluna (s) especificada (s) deve (m) ser único (s) para cada linha de uma tabela

PRIMARY KEY - valor (es) na (s) coluna (s) especificada (s) deve (m) ser única (s) para cada linha de uma tabela e não ser NULL; normalmente cada tabela em um banco de dados deve ter uma chave primária - ela é usada para identificar registros individuais

FOREIGN KEY - valor (es) na (s) coluna (s) especificada (s) deve referenciar um registro existente em outra tabela (via sua chave primária ou alguma outra restrição exclusiva)

CHECK - uma expressão é especificada, que deve ser avaliada como verdadeira para que a restrição seja satisfeita

DEFAULT – força um valor padrão quando nenhum valor é passado

Ex:

```
CREATE TABLE Canal(
CanalId INT PRIMARY KEY,
Nome VARCHAR(250) NOT NULL,
ContagemInscritos INT DEFAULT 0,
DataCriacao DATETIME NOT NULL
);


CREATE TABLE Video(
VideoId INT PRIMARY KEY,
Nome VARCHAR(150) NOT NULL,
Vizualizacoes INT DEFAULT 0,
Likes INT DEFAULT 0,
```

```
Dislikes INT DEFAULT 0,
Duracao INT DEFAULT 0,
CanalId INT FOREIGN KEY REFERENCES Canal(CanalId)
);
```

19. **INSERT INTO – Usado par inserir dados em uma tabela criada usando os comando insert into, sempre usar a mesma quantidade de dados entre coluna e valores ao menos se a coluna tiver restrição de valores default**

Ex:

INSERT INTO nomeTabela(coluna1, coluna2, ...)

VALUES(valor1, valor2),

(valor1, valor2),

(valor1, valor2);

Ex:

INSERT INTO TabelaA (coluna1)

SELECT coluna2

FROM TabelaB

Ex:

```
INSERT INTO aula(id, nome)
VALUES
(1, 'aula1'),
(2, 'aula2'),
(3, 'aula3'),
(4, 'aula4');
```

Ex:

```
-- Para criar uma nova com os mesmos dados da tabela referenciada
SELECT *
INTO tabelaNova
FROM aula
```

20. **UPDATE – Comando usado para alterar linhas de uma tabela dentro do banco de dados, pode ser alterado uma linha como todas as linhas da tabela informada, (sempre coloque um WHERE, se não será alterado todas as linhas com aquele valor informado).**

Ex:

UPDATE nomeTabela

SET coluna1 = valor1

Coluna 2 = valor2

WHERE condicao

Ex:

```
UPDATE aula
SET nome = 'teste'
WHERE id = 1;
```

21. **DELETE – Usado para apagar linhas do banco de dados (sempre coloque um WHERE, se não irá apagar todas as linhas).**

Ex:

DELETE FROM nomeTabela

WHERE condição

Ex:

```
DELETE FROM aula
WHERE id = 3;
```

22. **ALTER TABLE – Usado para alterar a estrutura de uma tabela.**

Exemplo do que pode ser feito:

- Add, Remover, ou alterar uma coluna

- Setar valores padrões para uma coluna

- Add ou Remover restrições de colunas

- Renomear uma tabela

Ex:

ALTER TABLE nomeTabela

ACAO

Ex:

```
--ADD -> Usado para adicionar colunas em uma tabela existente
ALTER TABLE YouTube
ADD ativo bit

-- ALTER COLUMN -> Usado para alterar (tipo ou restrição) de uma coluna
ALTER TABLE Youtube
ALTER COLUMN categoria VARCHAR(300) NOT NULL

-- EXEC sp_RENAME -> Usado para renomear uma coluna/tabela
EXEC sp_RENAME 'nomeTabela.nomeColunaAtual', 'nomeColunaNova', 'COLUMN'
-- Renomeando uma coluna
EXEC sp_RENAME 'YouTube.nome', 'nomeCanal', 'COLUMN'
-- Renomeando uma tabela
EXEC SP_RENAME 'YouTube', 'Youtube2'
```

23. **DROP TABLE – Comando usado para dropar (excluir), uma tabela do banco de dados (a tabela não pode ser referenciada por outra tabela).**

Ex:

DROP TABLE nomeTabela

Ex:

```
DROP TABLE CasaCheiaDeFrufru
```

24. **TRUNCATE TABLE – Comando usado para excluir o conteúdo (valores) de uma tabela.**

EX:

TRUNCATE TABLE nomeTabela

EX:

```
TRUNCATE TABLE teste2
```

25. **CHECK CONSTRAINT – Usado para criar restrições de valores que podem ser inseridos em uma coluna de uma tabela em banco de dados quando você está criando uma tabela nova.**

Ex:

```
CREATE TABLE CarteiraMotorista(
Id INT NOT NULL,
Nome VARCHAR (255) NOT NULL,
Idade INT CHECK ( Idade >= 18 )
);
```

Ex (Erro):

```
-- Teste de criação de valores que não vai ser aceito por Rafael ser menor de 18
anos
INSERT INTO CarteiraMotorista(Id, Nome, Idade)
VALUES
(1, 'Rafael', 17);

-- Teste de criação de valores que será aceito por João ser maior de 18 anos
INSERT INTO CarteiraMotorista(Id, Nome, Idade)
VALUES
(1, 'João', 18);
```

26. **NOT NULL – Usado para criar restrições, forçam com que não seja possível inserir dados em uma tabela sem preencher está coluna marcar como NOT NULL.**

EX:

```
CREATE TABLE CarteiraMotorista(
Id INT NOT NULL,
Nome VARCHAR (255) NOT NULL,
Idade INT CHECK ( Idade >= 18 )
);
```

Ex (Erro):

```
-- Teste de criação de valores que não será aceito por não conter o nome
INSERT INTO CarteiraMotorista(Id, Nome, Idade)
VALUES
(2, , 18);
```

Ex:

```
-- Teste de criação de valores que vai ser aceito por conter as validações
corretas do NOT NULL
INSERT INTO CarteiraMotorista(Id, Nome, Idade)
VALUES
(1, 'Rafael', 18);
```

27. **UNIQUE CONSTRAINT – Usado para ter colunas com valores únicos (SEM DADOS REPETIDOS), porém diferente de uma primary key, pode existir várias colunas com a restrição UNIQUE.**

Ex:

```
CREATE TABLE CarteiraMotorista(
Id INT NOT NULL,
Nome VARCHAR (255) NOT NULL,
Idade INT CHECK ( Idade >= 18 ),
CodigoCNH INT NOT NULL UNIQUE
);
```

EX(Erro):

```
Mensagem 2627, Nível 14, Estado 1, Linha 5
Violation of UNIQUE KEY constraint 'UQ__Carteira__F42C573F3929AAE3'. Cannot
insert duplicate key in object 'dbo.CarteiraMotorista'. The duplicate key value
is (123456).
The statement has been terminated.
```

```sql
-- UNIQUE -> Não será válido por na coluna informada já tem o valor em outra
linha da tabela
INSERT INTO CarteiraMotorista(id, Nome, Idade, CodigoCNH)
VALUES (2, 'Rafael', 18, 123456)
```

Ex:

```sql
-- UNIQUE -> Será válido por na coluna informada não tem o valor em outra linha
da tabela
INSERT INTO CarteiraMotorista(id, Nome, Idade, CodigoCNH)
VALUES (2, 'Rafael', 18, 1234567)
```

28. **VIEWS – Comando que são tabela criadas para consulta onde você usa outras tabelas como base para criar uma nova tabela de pesquisa com apenas dados específicos que você precisa de uma tabela.**

Ex:

```sql
-- VIEW -> Criando a view
CREATE VIEW [Pessoas Simplificado] AS
SELECT FirstName, MiddleName, LastName
FROM person.Person
WHERE Title = 'Ms.'
```

Ex:

```sql
-- VIEW -> Chamando a view criada acima
SELECT *
FROM [Pessoas Simplificado]
```