INTRODUÇÃO AO SQL SERVER

Selecionando apenas as primeira X linhas ou %X linhas:
SELECT TOP(5) countries FROM database;
SELECT TOP(5) PERCENT PERCENT countries FROM database;

Selecionando apenas valores distintos:
SELECT DISTINCT countries FROM database;

Selecionando todos os valores:
SELECT * FROM database;

Selecionando e escolhendo o nome da coluna:
SELECT countries as paises FROM database;

Selecionando mais de uma coluna:
SELECT countries, states FROM database;

Ordenando de acordo com algo:
SELECT * FROM database ORDER BY year; //também é possível ordenar strings

Ordenando de acordo com algo em descendente:
SELECT * FROM database ORDER BY year DESC, produto;

Condição de seleção:
SELECT countries, states FROM database WHERE population > 50000;
// para strings: WHERE countries = 'JAPAO'
// para datas: WHERE date = '2013-12-22''

Condição de seleção "entre":
SELECT countries, states FROM database WHERE population BETWEEN 50000 and 100000;
// negação: NOT BETWEEN 10 and 20;

Selecionando null's:
SELECT countries, states FROM database WHERE city IS NULL;
// não selecionando: WHERE city IS NOT NULL;

Condição com AND e OR:
SELECT song, artist FROM playlist WHERE artist = 'AC/DC' AND release_year <= 1990;
// ou OR
// pode-se usar mais de um AND: WHERE artist = 'AC/DC' AND (release _year <= 1990 OR name = 'hello');

Condição "em":
SELECT song, release_year FROM songlist WHERE release_year IN (1982,1986,1990,1992);

Condição com busca de string:
SELECT song FROM songlist WHERE song LIKE 'a%'; // sons começados com 'a'
// para terminar em a: LIKE '%a';

OPERAÇÕES COM T-SQL

Calculando o valor total de uma coluna:

```
SELECT SUM(albums_sold) AS total_sold FROM songlist;
// lembrando que caso se queira selecionar outra coluna, ela também precisa ser uma função de agregação
Ex.
SELECT
        SUM(albums_sold) AS total_sold,
        SUM(songs_played) AS total _plays
 FROM songlist;
```

Contando a quantidade de valores em uma coluna:

```
SELECT
        COUNT(affected_customers) AS count_affected
FROM grid;
```

Contando a quantidade de valores distintos em uma coluna:

```
SELECT
        COUNT(DISTINCT affected_customers) AS count_affected
FROM grid;
```

Buscando o valor mínimo:

```
SELECT
        MIN(affected_customers) AS min_affected_customers
FROM grid;
// Escolhendo o mínimo excluindo o zero
SELECT
        MIN(affected_customers) AS min_affected_customers
FROM grid
WHERE affected_customers > 0;
```

Buscando o valor máximo:

```
SELECT
        MAX(affected_customers) AS max_affected_customers
FROM grid;
```

Buscando o valor médio:

```
SELECT
        AVG(affected_customers) AS avg_affected_customers
FROM grid;
```

Contando o número de caracteres de uma string:

```
SELECT
        LEN(song_names) AS songs_length
FROM grid;
```

Selecionando valores distintos

```
SELECT DISTINCT * FROM TABELA;     //retorna dados únicos diferentes
SELECT DISTINCT * EMBALAGEM, TAMANHO FROM tabela_produtos WHERE sabor = ‘Laranja’;
```

Selecionando apenas um pedaço da string:

```
SELECT
        LEFT(song_names) AS songs_13_left
FROM grid;
// output: eye of the ti  em vez  de eye of the tiger
// para pegar os últimos caracteres: RIGHT(song_names)
```

Buscando por algo em uma string “até algo”:

```
SELECT
        CHARINDEX(‘_’, url) AS char_location,
        url
FROM courses;
// output: datacamp.com/courses/writting_
```

Selecionando N caracteres a partir da posição X:

```
SELECT
        SUBSTRING(url, 13, 14) AS target_location,
        url
FROM courses;
```

Trocando uma letra por outra:

```
SELECT
        REPLACE(url, “_”, “-”) AS replace_within_hyphen
FROM courses;
// output: datacamp.com/courses/writting-
```

Limitando a saída da consulta

```
 SELECT * FROM tabela_produtos LIMIT 4;     //limit fica sempre no final da consulta
SELECT * FROM TABELA LIMIT 2,3;             //a  partir do segundo,seleciona os 3 próximos
```

Agrupando valores:

```
SELECT
        SUM(demand_loss_nw) AS lost_demand,
        description
FROM grid
GROUP BY description;
```

Agrupando valores restringindo uma condição:

```
SELECT
        SUM(demand_loss_nw) AS lost_demand,
        description
FROM grid
WHERE
        description LIKE ‘%storm’
        AND demand_loss_mw IS NOT NULL
GROUP BY description;
HAVING SUM(demand_loss_mw) > 1000;
```

Chave primária

-Identifica unicamente cada linha de uma coluna, ou seja, não poderá conter valores nulos ou repetidos;
-Normalmente são identificadas com um ‘id’ na nomenclatura, como album_id;
-Caso haja duas chaves primárias, uma delas pode se repetir, mas a combinação delas não pode ser repetida. Ex: 5,9 e 5,7 (OK); 5,9 e 5,9 (INVÁLIDO).

Ex.

```
CREATE TABLE Pessoa
(
   ID_Pessoa integer PRIMARY KEY AUTOINCREMENT,
   Nome varchar(255),
   Endereco varchar(255),
   Cidade varchar(255)
);
```

Chaves estrangeiras

-Ou chave externa;
-Relacionamento entre tabelas distintas do banco de dados;
-“Uma chave estrangeira é um campo, que aponta para a chave primária de outra tabela ou da mesma tabela. Ou seja, passa a existir uma relação entre duplas de duas tabelas ou de uma única tabela. A finalidade da chave estrangeira é garantir a integridade dos dados referenciais, pois apenas serão permitidos valores que supostamente vão aparecer na base de dados. Esse tipo de atributo não permite exclusão, modificação ou inserção de dados em tabelas que estejam dependentes umas das outras ("foreign key"), o que requer modificadores especiais, como *cascade*, por exemplo.“
Ex.

```
CREATE TABLE Carro
(
   ID_Carro integer PRIMARY KEY AUTOINCREMENT,
   Nome varchar(255),
   Marca varchar(255),
   ID_Pessoa integer,
   CONSTRAINT fk_PesCarro FOREIGN KEY (ID_Pessoa) REFERENCES
Pessoa(ID_Pessoa)
);
```

Join de colunas

-Une uma ou mais tabelas através de campos em comum -INNER JOIN, LEFT JOIN , FULL JOIN, CROSS JOIN

INNER JOIN

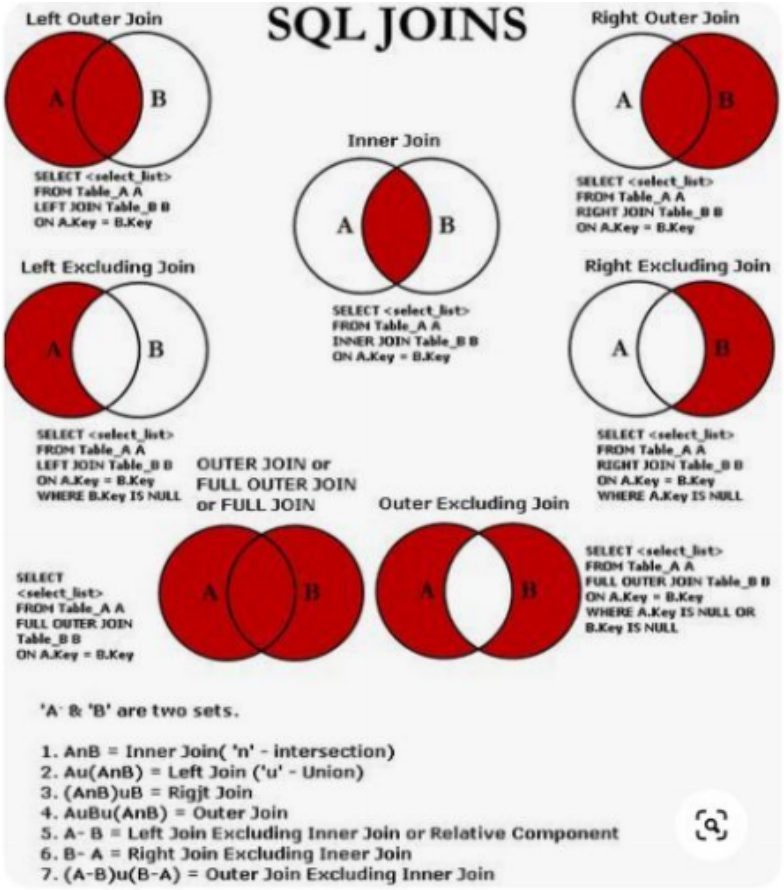
-Retorna os valores iguais entre duas tabelas
Ex.

```
SELECT
        A.MATRICULA, A.NOME
        , COUNT(*)
FROM tabela_vendedores A
INNER JOIN
        notas_fiscais B ON A.MATRICULA = B.MATRICULA
GROUP BY A.MATRICULA, A.NOME;
```

LEFT/RIGHT JOIN

-Traz todo mundo da tabela da esquerda/direta e apenas os correspondentes da tabela da direita/esquerda.
Ex.procurar quem nunca comprou na loja:

```
SELECT DISTINCT
        A.CPF, A.NOME,
        B.CPF FROM tabela_de_clientes A
LEFT JOIN notas_fiscais B on A.CPF = B.CPF
WHERE B.CPF IS NULL
```



Combinando resultados de buscas

```
SELECT
        album_id,
        title,
        artist_id
FROM album
WHERE artist_in IN (1, 3)
UNION                                   // não inclui linhas repetidas
SELECT                                  // colunas precisam ter mesmo tipo e ordem
        album_id,
        title,
        artist_id
FROM album
WHERE artist_in IN (1, 4, 5)
```

Combinando todos os resultados de busca

```
SELECT
        album_id,
        title,
        artist_id
FROM album
WHERE artist_in IN (1, 3)
UNION ALL                               // inclui linhas repetidas
SELECT
        album_id,
        title,
        artist_id
FROM album
WHERE artist_in IN (1, 4, 5)
```

Operadores CRUD

CREATE - Databases, Tables, Views; User, permissions e security groups
>>> CREATE TABLE unique table name (column name, data type, size)
Ex.

```
CREATE TABLE test_table(                    // IF NOT EXISTS
        test_data date,
        test_name varchar(20),
        test_int int
)
```

READ - Selecione (Select) as declarações
UPDATE - Corrigir registros de banco de dados existentes
DELETE - Deletar registros

INSERT, UPDATE, DELETE

INSERT - inserir dados
>>> INSERT INTO table_name (col1, col2, col3) values ('value1', 'value2, value3)

```
// Inserindo dados 'buscados' de outra tabela
INSERT INTO table_name (col1, col2, col3)
SELECT
        column1,
        column2,
        column3
FROM other_table
WHERE
        -- conditions
```

UPDATE - Atualizar dados de uma tabela
Ex.

```
UPDATE table
SET
        column1 = value1,
        column2 = value2
WHERE
        -- conditions
```

DELETE - Deletar algo (lembre-se que não há confirmação)
Ex.

```
DELETE
FROM table
WHERE
        -- conditions
// para excluir dados de todas as colunas de uma vez
TRUNCATE TABLE table_name
```

Criando Variáveis

-Úteis para evitar repetições de código.
Ex. Para selecionar as informações de um artista (note a recomendação de utilizar um @ na frente do nome da variável):

```
SELECT *
FROM artist
WHERE name = @my_artist;
// depois disso, basta alterar o valor da variável @my_artist
Ex.
DECLARE @my_artist VARCHAR (100)
DECLARE @my_album VARCHAR (100);

SET @my_artist = 'AC/DC'
SET @my_album = 'Let There Be Rock'';
```

```
SELECT --
FROM --
WHERE artist = @my_artist
AND album = @my_album;
```

Tabelas temporárias

```
SELECT
        col1,
        col2,
        col3 INTO #my_temp_table        // existirá até a sessão ser encerrada ou ser excluída
FROM my_existing_table
WHERE
        -- conditions
```