Criando um banco de dados de nome DB_CURSO_2024

```
CREATE DATABASE DB_CURSO_2024;
```

Selecionando o Banco de dados DB_CURSO_2024, para se conectar ao banco

```
USE DB_CURSO_2024;
```

Deletando um banco de dados de nome DB_CURSO_2024

```
DROP DATABASE DB_CURSO_2024;
```

CRIANDO UMA TABELA

```
-- Informações que precisam constar na nossa tabela
-- CPF do cliente "O cpf sempre vai ocupar 11 digitos"
-- O nome completo do cliente
-- Endereço com Rua, bairro, cidade, estado e CEP
-- Data de nascimento
-- A idade
-- Gênero
-- O limite do crédito do cliente para ele comprar produtos na empresa
-- O volume mínimo de produtos que ele pode comprar
-- Se ele já realizou a primeira compra
```

OBSERVAÇÕES

CPF do Cliente -- O cpf sempre vai ocupar 11 dígitos
É comum, no entanto, que CPFs iniciem com o numeral 0, por isso, se colocamos 01, a leitura é 1. Para evitar esse problema, teríamos que colocá-lo como caractere, '01', mantendo, assim, o 0 inicial. Pensando nesta problemática, é necessário que coloquemos o campo CPF como texto, e não como número

Lembre-se que, diferente do CHAR, o tipo VARCHAR não completa os caracteres restantes com espaços em branco.

Faremos primeiro dois campos de "endereço", um para o nome da rua e outro para o complemento

O CEP, por sua vez, é um número de 8 dígitos, embora pudesse ser 9 caso quiséssemos considerar o traço, mas optaremos pelo tamanho (8). Como há CEPs que iniciam em 0, o definiremos como [CHAR], assim como no CPF

O limite de crédito refere-se a valores monetários e não requer um tamanho, então será do tipo [MONEY]

O último campo, que chamaremos de [PRIMEIRA COMPRA], quer saber se o cliente já realizou alguma compra, então trata-se de um campo lógico que definiremos como do tipo [BIT].

Podemos fazer a criação da tabela dessa maneira, utilizando []. O uso de [] é muitas vezes opcional, só devemos usar obrigatoriamente quando queremos que as palavras fiquem separadas por espaços

```
CREATE TABLE [TABELA DE CLIENTES](
[CPF] [CHAR] (11),
[NOME] [VARCHAR] (100),
[ENDERECO1] [VARCHAR] (150),
[ENDERECO2] [VARCHAR] (150),
[BAIRRO] [VARCHAR] (50),
[CIDADE] [VARCHAR] (50),
[ESTADO] [CHAR] (2),
[CEP] [CHAR] (8),
[DATA DE NASCIMENTO] [DATE],
[IDADE] [SMALLINT],
[GENERO] [CHAR] (1),
[LIMITE DE CREDITO] [MONEY],
[VOLUME DE COMPRA] [FLOAT],
[PRIMEIRA COMPRA] [BIT]
);
```

Recomenda se utilizar dessa maneira abaixo

```
CREATE TABLE TABELA_DE_CLIENTES(
CPF CHAR (11),
NOME VARCHAR (100),
ENDERECO1 VARCHAR (150),
ENDERECO2 VARCHAR (150),
BAIRRO VARCHAR (50),
CIDADE VARCHAR (50),
ESTADO CHAR (2),
CEP CHAR(8),
DATA_DE_NASCIMENTO DATE,
IDADE SMALLINT,
GENERO CHAR (1),
LIMITE_DE_CREDITO MONEY,
VOLUME_DE_COMPRA FLOAT,
PRIMEIRA_COMPRA BIT
);
```

CRIANDO A TABELA DE PRODUTOS

```
-- Código do produto
-- Nome do produto
-- Embalagem
-- Tamanho
-- Sabor
-- Preço de lista
```

```
CREATE TABLE TABELA_DE_PRODUTOS(
CODIGO_DO_PRODUTO VARCHAR (20),
NOME_DO_PRODUTO VARCHAR (150),
EMBALAGEM VARCHAR (50),
TAMANHO VARCHAR (50),
SABOR VARCHAR (50),
PRECO_DE_LISTA SMALLMONEY
);
```

Diferença de Money e Smallmoney

Tipo de dado:	Valor mínimo:	Valor máximo:	Armazenamento:
smallmoney	-214,748.3648	214,748.3647	4 bytes
money	-922,337,203,685,477.5808	922,337,203,685,477.5807	8 bytes

A regra é sempre usar o tipo de dados que vai exigir menos armazenamento.
Smallmoney é como money, porém menor. Especificamente money é um 8 bit Decimal, enquanto smallmoney é um 4 bit Decimal
Para atribuir o valor R$ 34.654,79 devemos fornecê-lo como 34654.78 ou seja substituindo a vírgula por um ponto e eliminando os pontos dos milhares.

CRIANDO A TABELA DE VENDEDORES

```
CREATE TABLE TABELA_DE_VENDEDORES(
MATRICULA CHAR(5),
NOME VARCHAR(100),
PERCENTUAL_COMISSAO FLOAT
);
```

Alterar a coluna "CPF" para que ela não aceite nulos.

```
ALTER TABLE TABELA_DE_CLIENTES ALTER COLUMN CPF CHAR(11) NOT NULL;
```

Criando uma chave primária.

```
ALTER TABLE TABELA_DE_CLIENTES ADD CONSTRAINT PK_TABELA_DE_CLIENTES
PRIMARY KEY CLUSTERED (CPF);
```

INSERINDO DADOS NA TABELA

Incluindo dados na tabela. Não colocaremos o nome dos campos, nós adicionaremos os dados na mesma ordem em que os campos estão declarados.

```
INSERT INTO TABELA_DE_CLIENTES
VALUES
('00384393431', 'João da Silva', 'Rua Projetada A1', 'Numero 233', 'Copacabana',
'Rio de Janeiro', 'RJ', '20000000', '1965-03-21', 57, 'M', 20000, 30000.30, 1);

INSERT INTO TABELA_DE_PRODUTOS (CODIGO_DO_PRODUTO, NOME_DO_PRODUTO, EMBALAGEM,
TAMANHO, SABOR, PRECO_DE_LISTA)
VALUES
('1040107','Light - 350 ml - Melância', 'Lata', '350 ml', 'Melância', 4.56);

INSERT INTO TABELA_DE_VENDEDORES (MATRICULA, NOME, PERCENTUAL_COMISSAO)
VALUES
('00235', 'Márcio Almeida Silva', 0.08);
```

Precisamos lembrar que o campo de comissão está no valor float, portanto, é relevante colocarmos 0.08 pois ali representaria a porcentagem (8%). Pensando em decimal, 1.00 seria 100%, por isso 0.08 é o valor que queremos para a comissão.

```
INSERT INTO TABELA_DE_PRODUTOS (CODIGO_DO_PRODUTO, NOME_DO_PRODUTO, EMBALAGEM,
TAMANHO, SABOR, PRECO_DE_LISTA)
VALUES
('1037797', 'Clean - 2 litros - Laranja', 'PET', '2 Litros', 'Laranja', 16.01 );

INSERT INTO TABELA_DE_PRODUTOS (CODIGO_DO_PRODUTO, NOME_DO_PRODUTO, EMBALAGEM,
TAMANHO, SABOR, PRECO_DE_LISTA)
VALUES
('1000889', 'Sabor da Montanha - 700 ml - Uva', 'Garrafa', '700 ml', 'Uva',6.31);
```

Quando as ordens dos campos forem respeitada, não é preciso declará-los no comando, podendo ficar desta forma:
Isso significa que que cada valor corresponde, respectivamente, a determinado campo da tabela

```
INSERT INTO TABELA_DE_PRODUTOS
VALUES
('1004327','Videira do Cmapo - 1,5 Litros - Melância','PET','1,5
Litros','Melância',19.51);
```

Esse INSERT acima e a mesma coisa que esse INSERT a seguir.

```
INSERT INTO TABELA_DE_PRODUTOS (CODIGO_DO_PRODUTO, NOME_DO_PRODUTO, EMBALAGEM,
TAMANHO, SABOR, PRECO_DE_LISTA)
VALUES
('1004327','Videira do Cmapo - 1,5 Litros - Melância','PET','1,5
Litros','Melância',19.51);
```

Esse INSERT abaixo, no campo "PRECO_DE_LISTA" inserimos somente 7 ao invés de 7.00, ainda assim não teve problema devido o tipo desse campo ser "smallmoney", ou seja, possui casas decimais.

```
INSERT INTO TABELA_DE_PRODUTOS
VALUES
('1088126','Linha Citros - 1 Litro - Limão','PET','1 Litro','Limão',7);
```

Dessa maneira abaixo estamos inserindo duas linhas na tabela ao mesmo tempo

```
INSERT INTO TABELA_DE_PRODUTOS
VALUES
('1088126','Linha Citros - 1 Litro - Limão','PET','1 Litro','Limão',7),
('544931','Frescor do Verão - 350 ml - Limão','Lata','350 ml','Limão',2.46);


INSERT INTO TABELA_DE_VENDEDORES (MATRICULA, NOME, PERCENTUAL_COMISSAO)
VALUES
('00236', 'Cláudia Morais', 0.08),
('00237', 'Marcela Ferreira', 0.09);
```

INCLUINDO VENDEDORES NA TABELA DE VENDEDORES

```
INSERT INTO TABELA_DE_VENDEDORES
(NOME,MATRICULA,CIDADE,PERCENTUAL_COMISSÃO,DATA_INICIO,TEM_DEPENDENTE)
VALUES
('João da Silva','00241','São Paulo',0.06,'2022-12-01','TRUE'),
('Carolina Soares','00242','Rio de Janeiro',0.07,'2020-03-13','TRUE'),
('Juliana Marques','00243','São Paulo',0.09,'2018-11-14','FALSE'),
('Pedro Gomes','00244','São Paulo',0.08,'2019-01-20','FALSE'),
('Christina Rodrigues','00245','Rio de Janeiro',0.07,'2021-02-13','TRUE');
```

REALIZANDO CONSULTAS

Comando para consultar uma tabela do banco de dados
Nesse caso quero visualizar todos os campos do jeito que estão declarados na tabela.
Pode ser realizado dessas duas maneiras.

```
SELECT * FROM TABELA_DE_PRODUTOS

SELECT * FROM [TABELA_DE_PRODUTOS]

SELECT  CPF, NOME, ENDERECO1, ENDERECO2, BAIRRO, CIDADE,
ESTADO, CEP, DATA_DE_NASCIMENTO, IDADE, GENERO,
LIMITE_DE_CREDITO, PRIMEIRA_COMPRA
FROM TABELA_DE_CLIENTES;
```

Quero que mostre na tela somente o campo/coluna "nome" e "cpf"

```
SELECT NOME, CPF * FROM TABELA_DE_CLIENTES;
```

Também podemos criar um Alias, ou seja, um Apelido. O Alias é um nome fantasia que daremos para a coluna. Por exemplo, após NOME, podemos escrever AS NOME_DO_CLIENTE, nesse caso "NOME_DO_CLIENTE" e o alias/apelido.

```
SELECT NOME AS NOME_DO_CLIENTE, CPF
FROM TABELA_DE_CLIENTES;

SELECT NOME AS NOME_DO_CLIENTE, CPF AS IDENTIFICADOR
 FROM TABELA_DE_CLIENTES;
```

APAGANDO UMA TABELA

```
DROP TABLE TABELA_DE_CLIENTES;
```

USANDO FILTROS NO UPDATE

```
USE SUCOS_VENDAS;

SELECT * FROM [TABELA DE PRODUTOS] WHERE [EMBALAGEM] = 'Lata';
```

Vamos supor que o executivo da empresa pediu que fizéssemos um reajuste de 10% em todos os produtos de lata, porque o custo do alumínio aumentou. Isto é, a lata ficou mais cara e por isso precisamos aumentar o preço desses produtos.
Para isso, podemos usar o UPDATE, já que ele altera um valor existente.
Para aumentar os preços dos produtos em 10%, será necessário multiplicar os valores do do campo "preco de lista" por 1.1
Podemos fazer o aumento de preço dos produtos, um por um dessa forma abaixo:

Nesse caso abaixo estou modificando o valor do produto de código 544931

```
UPDATE [TABELA DE PRODUTOS] SET [PRECO DE LISTA] = 4.555 * 1.1
WHERE [CODIGO DO PRODUTO] = '544931';
```

/*Aumentando o preço de todos os produtos de uma vez somente
Aumentando em 10%, o “preço de lista” de todos os produtos com embalagem de lata. Usando apenas um comando.

```
UPDATE [TABELA DE PRODUTOS] SET [PRECO DE LISTA] = [PRECO DE
LISTA] * 1.1 WHERE [EMBALAGEM] = 'Lata';
```

FILTROS COMPOSTOS

Os "Filtros compostos" acontecem quando reunimos mais de uma condição no mesmo filtro,
Por exemplo: visualizar todas as pessoas que nasceram no ano de 1995 e moram no Rio de Janeiro.
Neste caso, vamos juntar condições e usar dois símbolos lógicos: AND (E) e OR (OU).

As expressões com "AND" só serão verdadeiras quando as duas condições são verdadeiras.

VERDADEIRO **AND** VERDADEIRO --> VERDADEIRA
FALSO **AND** VERDADEIRO --> FALSA
VERDADEIRO **AND** FALSO --> FALSA
FALSO **AND** FALSO --> FALSA

O "OR" é um pouco diferente: se uma das condições for verdadeira, a expressão é verdadeira.

VERDADEIRO **OR** VERDADEIRO --> VERDADEIRA
FALSO **OR** VERDADEIRO --> VERDADEIRA
VERDADEIRO **OR** FALSO --> VERDADEIRA
FALSO **OR** FALSO --> FALSA

Realizar uma consulta para visualizar todos os campos da tabela de clientes referentes às pessoas que nasceram no dia 12.

```
SELECT * FROM [TABELA DE CLIENTES] WHERE DAY ([DATA DE NASCIMENTO]) = 12
```

Realizar uma consulta para visualizar todas as pessoas que nasceram no dia 12 "e"/AND moram no bairro da Tijuca.

```
SELECT * FROM [TABELA DE CLIENTES] WHERE DAY ([DATA DE NASCIMENTO]) =12 AND [BAIRRO] = 'Tijuca';
```

Realizar uma consulta para visualizar todas as pessoas que moram no bairro da Tijuca "ou"/OR no bairro dos Jardins.

```
SELECT * FROM [TABELA DE CLIENTES] WHERE [BAIRRO] = 'Jardins' OR [BAIRRO] = 'Tijuca';
```

UTILIZANDO FILTRO MAIOR, MENOR OU DIFERENTE

SINAIS
> MAIOR
< MENOR
<> DIFERENTE
>= MAIOR OU IGUAL
<= MENOR OU IGUAL

```
USE SUCOS_VENDAS

SELECT * FROM [TABELA DE PRODUTOS];
```

Se quisermos saber qual é o produto com preço de lista igual a 7,336

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] =
7.336;
```

Saber quais produtos custam mais que 7.336

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] >
7.336;
```

Saber quais produtos custam menos que 7.336

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] <
7.336;
```

Com a consulta abaixo, veremos todos os produtos que não custam 7,336

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] <>
7.004;
```

Com a consulta abaixo, vamos visualizar todos os produtos que custam mais que 7,336, incluindo os produtos que custam 7,336.

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] >=
7.336;
```

Com a consulta abaixo, vamos visualizar todos os produtos que custam menos que 7,336, incluindo os produtos que custam 7,336.

```
SELECT * FROM [TABELA DE PRODUTOS] WHERE [PRECO DE LISTA] <=
7.336;
```

FILTRANDO POR DATAS

Quando aplicamos maior, menor, maior igual, menor igual ou diferente, sobre textos, respeitamos a ordem alfabética e quando aplicamos sobre datas, respeitamos a ordem do calendário.

```
USE SUCOS_VENDAS;
GO
SELECT * FROM [TABELA DE CLIENTES];
GO
SELECT * FROM [TABELA DE CLIENTES]
WHERE [DATA DE NASCIMENTO] = '1995-09-11';
```

Estamos fazendo uma consulta, para descobrir quem nasceu depois da data de 11/09/1995. Com isso, visualizaremos todas as pessoas que nasceram depois do dia 11 de setembro de 1995.

```
SELECT * FROM [TABELA DE CLIENTES]
WHERE [DATA DE NASCIMENTO] > '1995-09-11';
```

Estamos fazendo uma consulta, para descobrir quem nasceu antes da data de 11/09/1995. Com isso, visualizaremos todas as pessoas que nasceram antes do dia 11 de setembro de 1995. Nosso resultado só tem datas menores que 11 de setembro de 1995.

```
SELECT * FROM [TABELA DE CLIENTES]
WHERE [DATA DE NASCIMENTO] < '1995-09-11';
```

Estamos fazendo uma consulta, para descobrir todas as pessoas que nasceram em 1995

```
SELECT * FROM [TABELA DE CLIENTES]
WHERE year ([DATA DE NASCIMENTO])= 1995;
```

Nós selecionamos os campos "Nome", "Estado", "Data de Nascimento" e o ano da "Data de nascimento". Por fim, apelidamos o campo " Data de nascimento" de "Ano"

```
SELECT [NOME], [ESTADO], [DATA DE NASCIMENTO], YEAR ([DATA DE NASCIMENTO]) AS ANO FROM [TABELA DE CLIENTES];
```

Nós selecionamos os campos "Nome", "Estado", "Data de Nascimento", ano da "Data de nascimento", mês da "Data de nascimento" e o dia da "Data de nascimento". Por fim, apelidamos o campo " Data de nascimento" de "Ano", Mês e dia.

```
SELECT [NOME], [ESTADO], [DATA DE NASCIMENTO],
 YEAR ([DATA DE NASCIMENTO]) AS ANO,
 MONTH ([DATA DE NASCIMENTO]) AS MES,
 DAY ([DATA DE NASCIMENTO]) AS DIA
FROM [TABELA DE CLIENTES];
```

Para visualizar todas as pessoas que nasceram em setembro, ou seja, mês 9, fizemos a seguinte consulta

```
SELECT [NOME], [ESTADO], [DATA DE NASCIMENTO]
, YEAR ([DATA DE NASCIMENTO]) AS ANO
, MONTH ([DATA DE NASCIMENTO]) AS MES
, DAY ([DATA DE NASCIMENTO]) AS DIA
FROM [TABELA DE CLIENTES]
WHERE MONTH ([DATA DE NASCIMENTO]) = 9;
```

DELETANDO LINHAS DE UMA TABELA

```
SELECT * FROM TABELA_DE_PRODUTOS;
```

Deletando registros ou linhas de uma tabela

```
DELETE FROM TABELA_DE_PRODUTOS
 WHERE CODIGO_DO_PRODUTO = '1078680';
```

ALTERANDO DADOS EM UMA TABELA

```
USE SUCOS_VENDAS;

SELECT * FROM TABELA_DE_PRODUTOS;
```

Fazendo a alteração do "preco de lista" do produto “Frescor do Verão - 350 ml - Limão" que atualmente custa 2,46, mas sofrerá uma alteração e passará para 3. Utilizando o comando dessa forma abaixo, ele não identificará a linha e mudará todos os preços de lista para 3. Precisamos criar um filtro.

```
UPDATE TABELA_DE_PRODUTOS
SET PRECO_DE_LISTA = 3;
```

Para que não seja alterado o valor do produto no campo "preco de lista" de todos os produtos, criei um filtro "WHERE" que utilizei para identificar o produto o campo "CODIGO_DO_PRODUTO"

```
UPDATE TABELA_DE_PRODUTOS
SET PRECO_DE_LISTA = 7.00
WHERE CODIGO_DO_PRODUTO = '544931';
```

Vamos fazer outras modificações, desta vez, no produto da linha 5: "Linha Citros - 1 Litro - Limão". A embalagem mudará de "PET" para "Lata" e o preço mudará de 7.00 para 7.50

```
UPDATE TABELA_DE_PRODUTOS
SET    PRECO_DE_LISTA = 7.50,
       EMBALAGEM = 'Garrafa'
WHERE CODIGO_DO_PRODUTO = '1088126';

SELECT * FROM TABELA_DE_VENDEDORES;

UPDATE TABELA_DE_VENDEDORES
SET PERCENTUAL_COMISSAO = 0.11
WHERE MATRICULA = '00235';

UPDATE TABELA_DE_VENDEDORES
SET NOME = 'Cláudia Morais Sousa'
WHERE MATRICULA = '00236';
```

ALTERANDO O CAMPO COLUNA DE UMA TABELA

ALTER TABLE e o comando utilizado para alterar uma tabela ou para alterar um campo da tabela

ALTER TABLE [TABELA DE PRODUTOS]

Poderíamos usar **ADD** se quiséssemos, por exemplo, adicionar uma nova coluna.

ALTER TABLE [TABELA DE PRODUTOS] ADD COLUMN

Porém, a coluna CODIGO_DO_PRODUTO já existe. Sendo assim, não usaremos **ADD**, mas **ALTER**, de "alterar".
Descrição do comando abaixo: Alterar a coluna "tabela de produtos" modificando a coluna/campo "codigo do produto" para o tipo "varchar(20)" e que não aceite valores nulos

```
ALTER TABLE TABELA_DE_PRODUTOS ALTER COLUMN
CODIGO_DO_PRODUTO VARCHAR(20) NOT NULL;
```

ALTERNATIVAS PARA O CAMPO LÓGICO - BIT

```
SELECT * FROM TABELA_DE_CLIENTES;

INSERT INTO TABELA_DE_CLIENTES
VALUES
    ('00384393555', 'Maria Clara', 'Rua Projetada A1', 'Numero 233',
'Copacabana', 'Rio de janeiro', 'RJ', '20000000', '1975-03-21', 47, 'F', 20000,
30000.30, 'TRUE');

INSERT INTO TABELA_DE_CLIENTES
VALUES
    ('00384399999', 'Bruno da Silva', 'Rua Projetada 29', 'Numero 273', 'Caricica',
'Rio de janeiro', 'RJ', '20050000', '1975-03-21', 62, 'M', 30000, 40000.30, 1);

INSERT INTO TABELA_DE_CLIENTES
VALUES
    ('00384393666', 'Márcia Pereira', 'Rua Projetada A1', 'Numero 233',
'Copacabana', 'Rio de janeiro', 'RJ', '20000000', '1975-03-21', 47, 'F', 20000,
30000.30, 'FALSE');
```

O campo "primeira_compra" é do tipo bit, podemos representá-lo tanto com 0 ou 1, quanto com True or False. Lembrando que esta segunda opção precisa estar escrita em letras maiúsculas e aspas simples.
Quando escrevemos a palavra "True" no campo bit, ele entende que significa verdadeiro e usa o número 1 na tabela.
Ao escrevermos "False" no comando, no banco, ele será representado por zero.

Então, no campo lógico bit, podemos usar os números “1” ou 0” ou as palavras ‘True’ ou ‘False’, entre aspas simples e em letras maiúsculas.

ADICIONANDO UMA CHAVE PRIMARIA EM UMA TABELA JA EXISTENTE

```
USE SUCOS_VENDAS;
```

Podemos ter dois produtos com o mesmo nome, embalagem, tamanho, sabor e até com o mesmo preço, mas o código tem que ser diferente. Precisamos dizer para o banco de dados que o campo "CODIGO DO PRODUTO"
agora será uma Primary key. Para isso, é necessário alterar a estrutura da tabela de produtos:

ALTER TABLE TABELA_DE_PRODUTOS.

Nossa intenção é adicionar uma chave primária à tabela existente.

Restrição(constraints)

Podemos ter uma chave primária com mais de um campo, mas cada tabela pode ter apenas uma chave primária.

Para adicionar uma chave primaria faremos "PRIMARY KEY CLUSTERED" entre parênteses, o(s) campo(s) que será chave primaria

```
ALTER TABLE TABELA_DE_PRODUTOS
ADD CONSTRAINT PK_TABELA_DE_PRODUTOS
PRIMARY KEY CLUSTERED (CODIGO_DO_PRODUTO);
```

Explicando o comando acima:
Alterando “ALTER TABLE" a tabela "TABELA_DE_PRODUTOS"
adicionando uma chave primaria chamada de "PK_TABELA_DE_PRODUTOS"
no campo "CODIGO_DO_PRODUTO"

ADICIONANDO UMA CHAVE PRIMARIA NA TABELA DE VENDEDORES

```
USE SUCOS_VENDAS;

SELECT * FROM TABELA_DE_VENDEDORES;
```

Alterar o campo/coluna para que aceite valores nulos.

```
ALTER TABLE TABELA_DE_VENDEDORES
ALTER COLUMN MATRICULA CHAR(5) NOT NULL;
```

Criar a chave primária

```
ALTER TABLE TABELA_DE_VENDEDORES
ADD CONSTRAINT PK_TABELA_DE_VENDEDORES
PRIMARY KEY CLUSTERED (MATRICULA);
```

DESAFIO- MOSTRANDO DADOS NA TABELA

```
USE SUCOS_VENDAS;

SELECT * FROM TABELA_DE_VENDEDORES;
```

Visualize a tabela [TABELA DE VENDEDORES] mostrando o campo MATRICULA com o alias IDENTIFICADOR e o campo NOME com o alias NOME DO VENDEDOR

```
SELECT MATRICULA AS IDENTIFICADOR, NOME AS
NOME_DO_VENDEDOR
FROM TABELA_DE_VENDEDORES;
```

Visualize a tabela [TABELA DE VENDEDORES] mostrando o campo MATRICULA com o alias IDENTIFICADOR e o campo NOME com o alias NOME DO VENDEDOR, mas apenas os que atuam em São Paulo.

```
SELECT MATRICULA AS IDENTIFICADOR, NOME AS
NOME_DO_VENDEDOR
FROM TABELA_DE_VENDEDORES WHERE CIDADE = 'São Paulo';
```

Para os vendedores que possuem dependentes aumente em 1 ponto percentual suas comissões.

```
UPDATE TABELA_DE_VENDEDORES
SET PERCENTUAL_COMISSÃO = PERCENTUAL_COMISSÃO + 0.01
WHERE TEM_DEPENDENTE = 1;
```

Mostre os vendedores que possuem comissão abaixo de 8%.

```
SELECT * FROM TABELA_DE_VENDEDORES WHERE
PERCENTUAL_COMISSÃO < 0.08;
```

Liste os vendedores que foram admitidos em 2020 e antes destes anos.

```
SELECT * FROM TABELA_DE_VENDEDORES
WHERE YEAR (DATA_INICIO) <= 2020;
```

Liste os vendedores que possuem dependentes e que estejam atuando na cidade do Rio de Janeiro.

```
SELECT * FROM TABELA_DE_VENDEDORES WHERE TEM_DEPENDENTE
= 1 AND CIDADE = 'Rio de Janeiro';
```

DESAFIO-TABELA-VENDEDORES

```
USE SUCOS_VENDAS;
```

```
SELECT * FROM TABELA_DE_VENDEDORES;
```

1° Passo: Apague a tabela existente.

```
DROP TABLE TABELA_DE_VENDEDORES;
```

2º Passo: Crie uma nova TABELA_DE_VENDEDORES. Algumas informações:

Nome da tabela deve ser TABELA DE VENDEDORES.
Vendedor tem como chave o número interno da matrícula (Nome do campo MATRICULA) que deve ser um texto de 5 posições que não pode ser NULL.
O nome do vendedor (Nome do campo NOME) deve ser um texto de 100 posições.
Cidade de atuação (Nome do campo CIDADE) deve ser um texto de 100 posições.
% de comissão. Este campo (Nome do campo PERCENTUAL COMISSÃO) representa quantos %
de comissão o vendedor ganha sobre cada venda e não pode ser NULL.
Crie um campo chamado DATA INICIO que será a data em que o vendedor começou a trabalhar na empresa.
Este campo não pode ser NULL.
Crie um campo lógico chamado TEM_DEPENDENTE que será TRUE se o vendedor possuir dependentes e FALSE se não possuir dependentes.

```
CREATE TABLE TABELA_DE_VENDEDORES(
   MATRICULA varchar(5) NOT NULL,
   NOME varchar(100) NULL,
   CIDADE varchar(100) NULL,
   PERCENTUAL_COMISSÃO FLOAT NOT NULL,
   DATA_INICIO date NOT NULL,
   TEM_DEPENDENTE BIT);
```

3º Passo: Inclua o vendedor Alberto de Sá Verneck, matrícula 00239, atua na cidade de São Paulo, com comissão de 8%, admitido em 05/06/2020 e possui dependentes.

```
INSERT INTO TABELA_DE_VENDEDORES (NOME, MATRICULA,
CIDADE,PERCENTUAL_COMISSÃO, DATA_INICIO,TEM_DEPENDENTE)
VALUES
('Alberto de Sá Verneck','00239','São Paulo', 0.08,'2020-06-05',1);
```

4° Passo: Inclua um novo vendedor usando no campo lógico a forma alternativa. Nome Marcela Almeida, matrícula 00240, atua no Rio de Janeiro, com comissão de 7%, admitido em 12/01/2021 e não possui dependentes.

```
INSERT INTO TABELA_DE_VENDEDORES (NOME, MATRICULA,
CIDADE,PERCENTUAL_COMISSÃO, DATA_INICIO,TEM_DEPENDENTE)
```

```
VALUES
('Marcela Almeida','00240','Rio de Janeiro', 0.07,'2021-01-12','FALSE');
```

5º Passo Visualize a tabela.

```
SELECT * FROM TABELA_DE_VENDEDORES;
```

CONSULTA SIMPLES DE UMA TABELA

```
USE SUCOS_FRUTAS;
```

Para visualizar todos os campos da tabela de clientes.

```
SELECT * FROM TABELA_DE_CLIENTES;
```

Visualizando somente os campos CPF, NOME, BAIRRO E CIDADE

```
SELECT CPF, NOME, BAIRRO, CIDADE
FROM TABELA_DE_CLIENTES;
```

Utilizando um "alias/apelido" no campo CPF, NOME, BAIRRO e CIDADE

```
SELECT CPF AS IDENTIFICADOR, NOME AS NOME_DE_CLIENTE,
BAIRRO, CIDADE
FROM TABELA_DE_CLIENTES;
```

Outra maneira de fazer a mesma consulta acima
Na consulta abaixo eu tenho nome de cliente com o espaço entre as palavras, porque eu coloquei aqui meu abre e fecha colchetes

```
SELECT CPF AS IDENTIFICADOR, NOME AS [NOME DE CLIENTE],
BAIRRO, CIDADE
FROM TABELA_DE_CLIENTES;
```

Dando um apelido/alias chamado "TDC" a tabela de clientes
Para o campo eu uso a palavra "AS", mas quando eu construo um alias para a tabela, eu tenho que suprimir/excluir o "AS".

```
SELECT CPF AS IDENTIFICADOR, NOME AS [NOME DE CLIENTE],
BAIRRO, CIDADE
FROM TABELA_DE_CLIENTES  TDC;
```

Outra forma de utilizar apelidos/aliás nas tabelas
Quando eu crio um alias para tabela, eu posso colocar o alias primeiro e depois o nome do campo, por exemplo:

```
SELECT TDC.CPF, TDC.NOME, TDC. BAIRRO
FROM TABELA_DE_CLIENTES TDC;
```

OBSERVAÇÃO:
Qual a diferença de usar com aliás e sem alias aqui?
Nesse caso nenhuma, mas nós vamos ver lá na frente que eu posso ter uma consulta que utilize mais de uma tabela e haja campos em comum nessas tabelas, então o alias nesse caso vai ser importante para eu poder diferenciar o campo que tem nome igual em tabelas diferentes.

E se eu quiser ver todo mundo, todos os campos, usando o asterisco e o alias, eu posso colocar assim:
Eu consigo olhar todos os campos da tabela TDC, porque eu estou usando aqui o alias

```
SELECT TDC.* FROM TABELA_DE_CLIENTES TDC;
```

eu não preciso, por exemplo, sempre usar o alias para diferenciar o campo de uma tabela ou outra, eu poderia, por exemplo, fazer isso daqui:
Aqui no caso eu não estou usando aliás, mas eu estou colocando o próprio nome da tabela como se fosse um alias.

```
SELECT TABELA_DE_CLIENTES.CPF, TABELA_DE_CLIENTES.NOME
FROM TABELA_DE_CLIENTES;
```

CONSULTA COM FILTRO

WHERE é como se fosse um filtro

Fazendo uma consulta e utilizando o filtro para que retorne somente o produto de código 290478

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE CODIGO_DO_PRODUTO = 290478;
```

Vai retornar somente os produtos que têm o sabor de laranja

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE SABOR = 'Laranja';
```

Consulta que vai retornar os produtos que a embalagem seja do tipo PET

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE EMBALAGEM = 'PET';
```

FILTROS QUANTITATIVOS

BETWEEN - Ver um valor entre um e outro

Vamos supor que eu queira ver somente os clientes que têm mais de 20 anos de idade

```
SELECT * FROM TABELA_DE_CLIENTES
```

```
 WHERE IDADE > 20;
```

ver quais são os clientes que têm menos de 20 anos de idade,

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE IDADE < 20;
```

Se eu quiser saber quantos clientes têm menos de 20 anos de idade, incluindo o 20, basta eu colocar o menor ou igual

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE IDADE <= 20;
```

Se eu quiser saber quantos clientes têm menos de 18 anos de idade, incluindo o 18, basta eu colocar o menor ou igual

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE IDADE < = 18;
```

ver quais são os clientes que têm menos de 18 anos de idade

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE IDADE < 18;
```

E se eu quiser ver todo mundo exceto os que têm 18 anos, ou seja, quero ver todo mundo menos quem tem 18 anos.

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE IDADE <> 18;
```

Para eu representar a data no SQL Server, basta eu escrever essa data no seguinte formato: ano, um traço, mês e um traço dia.
Sempre o mês com dois dígitos e o dia com dois dígitos, ou seja, se eu quiser colocar uma data que represente janeiro, no lugar do MM eu vou colocar 01, não posso colocar 1, 01.
Se for um dia, por exemplo, menor que 10, por exemplo, dia 8, eu tenho que colocar 08.

Vamos supor que eu queira saber todos os clientes que nasceram depois do dia 14 de novembro de 1995.

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE DATA_DE_NASCIMENTO >= '1995-11-14';
```

Se eu quiser ver todos que nasceram antes do dia 14, eu venho aqui e coloco o menor

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE DATA_DE_NASCIMENTO <= '1995-11-14';
```

Quero selecionar todo mundo na tabela de clientes onde o bairro seja maior ou igual que Lapa:
O que isso significa? Que todos os bairros cuja ordem alfabética começarem depois da palavra Lapa, vão aparecer nessa consulta.

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE BAIRRO >= 'Lapa';
```

EXPRESSÕES LOGICAS

```
USE SUCOS_FRUTAS;
```

OR = OU
AND = E

```
SELECT * FROM TABELA_DE_PRODUTOS;
```

eu quero saber qual produto tem o sabor manga e o tamanho 470 ml
Aqui nesse caso quero o resultado do produto que tenha o tamanho de 470 ml e ao mesmo tempo, seja do sabor manga

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR = 'Manga' AND TAMANHO = '470 ml';
```

Nessa consulta abaixo quero saber qual produto e do tamanho de 470 ml ou é de sabor manga.
Vai retornar um produto que tenha sabor de manga ou um produto que seja de tamanho 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE SABOR = 'Manga' OR TAMANHO = '470 ml';
```

Aqui vai mostrar todos os produtos que NÃO é do sabor de Manga e que sejam do tamanho de 470 ml
Nessa situação abaixo, o NOT está sendo aplicado somente no "SABOR = 'Manga'

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE NOT SABOR = 'Manga' AND TAMANHO = '470 ml';
```

Aqui vai mostrar todos os produtos que NÃO são do sabor de manga ou os produtos que tenham tamanho de 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE NOT SABOR = 'Manga' OR TAMANHO = '470 ml';
```

Nós colocamos aqui um abre e um fecha parênteses, para mostrar para o SQL que o NOT está sendo aplicado sobre esta expressão toda e não somente sobre a primeira parcela da expressão:

Aqui vai aparecer todos os produtos, menos os que sejam do tamanho 470 ml e ao mesmo tempo sejam do sabor de manga.

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE NOT (SABOR = 'Manga' AND TAMANHO='470 ml');
```

Aqui vai aparecer todos os produtos, menos os produtos que seja do sabor de manga ou que tenham o tamanho de 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE NOT (SABOR = 'Manga' OR TAMANHO='470 ml');
```

Comparando os resultados

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE NOT (SABOR = 'Manga' AND TAMANHO='470 ml');

SELECT * FROM TABELA_DE_PRODUTOS
WHERE NOT SABOR = 'Manga' AND TAMANHO = '470 ml';
```

Mostre na tela os produtos que seja do sabor de Manga OU sabor de Laranja OU sabor de Melancia

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE SABOR = 'Manga' OR SABOR = 'Laranja' OR SABOR = 'Melancia';
```

Quando eu tenho uma consulta com vários OR, igual essa consulta acima, posso fazer da seguinte maneira abaixo utilizando o IN
Essa consulta acima utilizando OR é igual a consulta abaixo utilizando IN

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR IN ('Manga', 'Laranja', 'Melancia');
```

Quero mostrar na tela todos os produtos que sejam do sabor manda “OU” laranja “OU” Melancia “E” que sejam do tamanho de 1 Litro

```
SELECT * FROM TABELA_DE_PRODUTOS
 WHERE (SABOR  in ('Manga', 'Laranja', 'Melancia')) AND TAMANHO = '1 Litro';
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro OU que morem na cidade de São Paulo

```
SELECT * FROM TABELA_DE_CLIENTES
 WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo');
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro “OU” que morem na cidade de São Paulo “E” que tenham a idade maior ou igual a 20 anos

```
SELECT * FROM TABELA_DE_CLIENTES WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo') AND IDADE >= 20;
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro “OU” que morem na cidade de São Paulo “E” que tenham idade entre 20 a 25

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo')
AND (IDADE >= 20 AND IDADE <= 25);
```

Também é possível utilizar o BETWEEN, pois ele permite eu fazer a mesma consulta de cima, ou seja, vou ter os mesmos resultados

```
SELECT * FROM TABELA_DE_CLIENTES
WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo')
AND (IDADE BETWEEN 20 AND 25);
```

USANDO O LIKE

```
USE SUCOS_FRUTAS;
```

Vamos selecionar dois sabores específicos da nossa tabela de produtos: Sabor Lima/Limão “OU” Sabor Morango/Limão

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR IN ('Lima/Limao', 'Morango/Limao');
```

Eu poderia fazer essa mesma consulta acima da seguinte maneira:
Vai retornar todos os produtos que tenham sabores que terminam com a palavra Limão

```
SELECT  * FROM TABELA_DE_PRODUTOS
WHERE SABOR LIKE '%Limao';
```

/*Estou fazendo uma consulta que retorne todos os produtos de sabores que terminem com a palavra Maca*/

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR LIKE '%Maca';
```

/*Estou fazendo uma consulta que retorne todos os produtos de sabores que iniciem com a palavra Morango*/

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR LIKE 'Morango%';
```

/*Estou fazendo uma consulta que retorne todos os produtos de sabores que iniciem com a palavra Morango e que sejam do tamanho PET*/

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE (SABOR LIKE 'Morango%') AND (EMBALAGEM = 'PET');

--consulta que diga quantos clientes possuem o sobrenome Silva.
--Para mim não importa o nome que vem antes de 'Silva' e nem o nome que
vem depois de 'Silva'
SELECT * FROM TABELA_DE_CLIENTES
 WHERE  NOME LIKE '%Silva%';
```

USANDO O DISTINCT

O SELECT DISTINCT instrução é usada para retornar apenas valores distintos (diferentes).
Dentro de uma tabela, uma coluna geralmente contém muitos valores duplicados e às vezes você deseja apenas listar os valores diferentes (distintos).
Trata-se de uma cláusula para eliminar repetições em consultas

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_PRODUTOS;

/*Essa consulta vai retornar o campo embalagem da tabela de produtos*/

SELECT EMBALAGEM FROM TABELA_DE_PRODUTOS;

/*Saber quantas embalagens diferentes tem meus produtos*/

SELECT DISTINCT EMBALAGEM FROM TABELA_DE_PRODUTOS;
```

Vai retornar todos os tipos de embalagens que o produto de sabor maça tem

```
SELECT DISTINCT EMBALAGEM FROM TABELA_DE_PRODUTOS
 WHERE SABOR = 'Maca';


SELECT DISTINCT EMBALAGEM, SABOR FROM TABELA_DE_PRODUTOS;
```

Executando uma consulta para obter o número de sabores existentes

```
SELECT DISTINCT SABOR FROM TABELA_DE_PRODUTOS;
```

LIMITANDO A SAIDA DA CONSULTA

O comando **TOP** mostra os primeiros registros de uma tabela.

O comando TOP limita sempre, mostrando as primeiras linhas daquela seleção que você está aplicando.

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_CLIENTES;
```

Dessa maneira vai mostrar na tela os 4 primeiros clientes

```
SELECT TOP 4 * FROM TABELA_DE_CLIENTES;

SELECT * FROM TABELA_DE_PRODUTOS;
```

Quero que mostre na tela somente os 5 primeiros produtos que estão cadastrados na tabela de produtos

```
SELECT TOP 5 * FROM TABELA_DE_PRODUTOS;
```

Quero que mostre na tela somente os 2 primeiros produtos que estão cadastrados na tabela de produtos e que sejam do sabor Maça

```
SELECT TOP 2 * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Maca';

SELECT * FROM NOTAS_FISCAIS;
```

Listando as 10 primeiras vendas do dia 01/10/2017

```
SELECT TOP 10 * FROM NOTAS_FISCAIS WHERE DATA_VENDA = '2017-10-01';
```

CONSULTAS AVANÇADAS

O comando SP_HELP mostra a estrutura da tabela, seus atributos, relacionamentos e, por fim, se essa tabela possui índice ou não

```
SP_HELP OPERACOES
GO
SP_HELP OPERACOES_BCP
GO
SP_HELP OPERACOES_INCREMENTAL
GO
SP_HELP ENVIO_OPERACOES_GBD

SELECT @@SERVERNAME,
        CONNECTIONPROPERTY('net_transport') AS net_transport,
        CONNECTIONPROPERTY('protocol_type') AS protocol_type,
        CONNECTIONPROPERTY('auth_scheme') AS auth_scheme,
        CONNECTIONPROPERTY('local_net_address') AS local_net_address,
        CONNECTIONPROPERTY('local_tcp_port') AS local_tcp_port,
        CONNECTIONPROPERTY('client_net_address') AS client_net_address
```

Listar os bancos de dados que residem em uma instância do SQL Server

```
SP_DATABASES;
```

CONSULTA SIMPLES DE UMA TABELA

```
USE SUCOS_FRUTAS;
```

Para visualizar todos os campos da tabela de clientes.

```
SELECT * FROM TABELA_DE_CLIENTES;
```

Visualizando somente os campos CPF, NOME, BAIRRO E CIDADE da tabela de CLIENTES

```
SELECT CPF, NOME, BAIRRO, CIDADE FROM TABELA_DE_CLIENTES;
```

UTILIZANDO APELIDO/ALIAS EM CAMPOS E TABELAS

Utilizando um "Alias/Apelido" no campo CPF, NOME

```
SELECT CPF AS IDENTIFICADOR, NOME AS NOME_DE_CLIENTE,
BAIRRO, CIDADE FROM TABELA_DE_CLIENTES;
```

Outra maneira de fazer a mesma consulta acima.
Na consulta abaixo eu tenho NOME DE CLIENTE com o espaço entre as palavras, porque eu coloquei "[ ] colchetes"

```
SELECT CPF AS IDENTIFICADOR, NOME AS [NOME DE CLIENTE],
BAIRRO, CIDADE FROM TABELA_DE_CLIENTES;
```

Dando um Apelido/Aliás chamado "TDC" a tabela de clientes
Para utilizar aliás em um campo uso a palavra "AS", mas quando eu construo um Aliás para a tabela, eu tenho que Suprimir/Excluir o "AS".
Quando vou dar um APELIDO a uma TABELA não coloco o "AS"

```
SELECT CPF AS IDENTIFICADOR, NOME AS [NOME DE CLIENTE],
BAIRRO, CIDADE FROM TABELA_DE_CLIENTES  TDC;
```

Outra forma de utilizar "Apelidos/Alias" nas tabelas.
Quando eu crio um apelido para a tabela, eu posso colocar o aliás e depois o nome do campo. Por exemplo: TDC é o Aliás, BAIRRO é o campo

```
SELECT TDC.CPF, TDC.NOME, TDC. BAIRRO FROM
TABELA_DE_CLIENTES TDC;
```

OBSERVAÇÃO:
Qual a diferença de usar com aliás e sem alias?

Nesse caso nenhuma, mas nós vamos ver lá na frente que eu posso ter uma consulta que utilize mais de uma tabela e haja campos em comum nessas tabelas, então o alias nesse caso vai ser importante para eu poder diferenciar o campo que tem nome igual em tabelas diferentes.

Se eu quiser ver todos os campos, usando o asterisco e o alias, eu posso colocar assim:
Eu consigo olhar todos os campos da tabela TDC, porque eu estou usando aqui o alias

```
SELECT TDC.* FROM TABELA_DE_CLIENTES TDC;
```

Não preciso sempre usar o alias para diferenciar o campo de uma tabela ou outra, eu poderia fazer isso daqui:
Aqui no caso não estou usando alias, mas estou colocando o próprio nome da tabela como se fosse um alias.

```
SELECT TABELA_DE_CLIENTES.CPF, TABELA_DE_CLIENTES.NOME
FROM TABELA_DE_CLIENTES;
```

CONSULTA COM FILTRO

WHERE é como se fosse um filtro

Fazendo uma consulta e utilizando o filtro para que retorne somente o produto de código 290478

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE
CODIGO_DO_PRODUTO = 290478;
```

Vai retornar somente os produtos que têm o sabor de laranja

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Laranja';
```

Consulta que vai retornar os produtos que a embalagem seja do tipo PET

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE EMBALAGEM = 'PET';
```

FILTROS QUANTITATIVOS

BETWEEN - Ver um valor entre um e outro

Vamos supor que eu queira ver somente os clientes que têm mais de 20 anos de idade

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE > 20;
```

Ver quais são os clientes que têm menos de 20 anos de idade

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE < 20;
```

Se eu quiser saber quantos clientes têm menos de 20 anos de idade, incluindo o 20, basta eu colocar o menor ou igual

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE <= 20;
```

Se eu quiser saber quantos clientes têm menos de 18 anos de idade, incluindo o 18, basta eu colocar o menor ou igual

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE < = 18;
```

Ver quais são os clientes que têm menos de 18 anos de idade

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE < 18;
```

Se eu quiser ver todo mundo exceto os que têm 18 anos, ou seja, quero ver todo mundo menos quem tem 18 anos

```
SELECT * FROM TABELA_DE_CLIENTES WHERE IDADE <> 18;
```

Para eu representar a data no SQL Server, basta eu escrever essa data no seguinte formato:
ano, um traço, mês e um traço dia.
2000-02-10
Sempre o mês com dois dígitos e o dia com dois dígitos, ou seja, se eu quiser colocar uma data que represente janeiro, no lugar do MM eu vou colocar 01, não posso colocar 1, 01.
Se for um dia, por exemplo, menor que 10, por exemplo, dia 8, eu tenho que colocar 08.

Vamos supor que eu queira saber todos os clientes que nasceram depois do dia 14 de novembro de 1995.

```
SELECT * FROM TABELA_DE_CLIENTES WHERE DATA_DE_NASCIMENTO
>= '1995-11-14';
```

Se eu quiser ver todos que nasceram antes do dia 14, eu venho aqui e coloco o menor

```
SELECT * FROM TABELA_DE_CLIENTES WHERE DATA_DE_NASCIMENTO
<= '1995-11-14';
```

Quero selecionar todo mundo na tabela de clientes onde o bairro seja maior ou igual que Lapa:
O que isso significa?
Que todos os bairros cuja ordem alfabética começarem depois da palavra Lapa, vão aparecer nessa consulta.

```
SELECT * FROM TABELA_DE_CLIENTES WHERE BAIRRO >= 'Lapa';
```

EXPRESSÕES LÓGICAS

```
USE SUCOS_FRUTAS;
```

OR OU
AND E

```
SELECT * FROM TABELA_DE_PRODUTOS;
```

Quero saber qual produto tem o sabor manga e o tamanho 470 ml
Aqui nesse caso quero o resultado do produto que tenha o tamanho de 470 ml e ao mesmo tempo, seja do sabor manga

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Manga' AND
TAMANHO = '470 ml';
```

Nessa consulta abaixo quero saber qual produto e do tamanho de 470 ml ou é de sabor manga.
Vai retornar um produto que tenha sabor de manga ou um produto que seja de tamanho 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Manga' OR
TAMANHO = '470 ml';
```

Aqui vai mostrar todos os produtos que não do sabor de Manga e que sejam do tamanho de 470 ml
Nessa situação abaixo, o NOT está sendo aplicado somente no "SABOR = 'Manga'"

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT SABOR = 'Manga'
AND TAMANHO = '470 ml';
```

Aqui vai mostrar todos os produtos que não são do sabor de manga ou os produtos que tenham tamanho de 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT SABOR = 'Manga'
OR TAMANHO = '470 ml';
```

Nós colocamos aqui um abre e um fecha parênteses, para mostrar para o SQL que o NOT está sendo aplicado sobre esta expressão toda e não somente sobre a primeira parcela da expressão:

Aqui vai aparecer todos os produtos, menos os que sejam do tamanho 470 ml e ao mesmo tempo sejam do sabor de manga

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT (SABOR = 'Manga' AND TAMANHO='470 ml');
```

Aqui vai aparecer todos os produtos, menos os produtos que seja do sabor de manga ou que tenham o tamanho de 470 ml

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT (SABOR = 'Manga' OR TAMANHO='470 ml');
```

Comparando os resultados:

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT (SABOR = 'Manga' AND TAMANHO='470 ml');
```

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOT SABOR = 'Manga' AND TAMANHO = '470 ml';
```

Mostre na tela os produtos que seja do sabor de Manga OU sabor de Laranja OU sabor de Melancia

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Manga' OR SABOR = 'Laranja' OR SABOR = 'Melancia';
```

Quando eu tenho uma consulta com vários OR, igual essa consulta acima, posso fazer da seguinte maneira
Tanto essa consulta abaixo, quanto essa consulta acima são iguais

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR IN ('Manga', 'Laranja', 'Melancia');
```

Quero mostrar na tela todos os produtos que sejam do sabor Manga OU Laranja OU Melancia E que sejam do tamanho de 1 Litro

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE (SABOR  in ('Manga', 'Laranja', 'Melancia')) AND TAMANHO = '1 Litro';
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro OU que morem na cidade de São Paulo

```
SELECT * FROM TABELA_DE_CLIENTES WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo');
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro OU que morem na cidade de São Paulo E que tenham a idade maior ou igual a 20 anos

```
SELECT * FROM TABELA_DE_CLIENTES WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo') AND IDADE >= 20;
```

Quero que mostre na tela todos os clientes que morem na cidade do Rio de Janeiro ou que morem na cidade de São Paulo E que tenham idade entre 20 a 25

```
SELECT * FROM TABELA_DE_CLIENTES WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo') AND (IDADE >= 20 AND IDADE <= 25);
```

-Também e possível utilizar o BETWEEN, pois ele permite eu fazer a mesma consulta de cima, ou seja, vou ter os mesmos resultados

```
SELECT * FROM TABELA_DE_CLIENTES WHERE CIDADE IN ('Rio de Janeiro', 'Sao Paulo') AND (IDADE BETWEEN 20 AND 25);
```

USANDO O LIKE

```
USE SUCOS_FRUTAS;
```

Vamos selecionar dois sabores específicos da nossa tabela de produtos:

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE SABOR IN ('Lima/Limao', 'Morango/Limao');
```

Eu podia fazer essa mesma consulta acima da seguinte maneira:
Vai retornar todos os produtos que tenham sabores que terminam com a palavra “Limão”

```
SELECT  * FROM TABELA_DE_PRODUTOS
WHERE SABOR LIKE '%Limao';
```

Estou fazendo uma consulta que retorne todos os produtos de sabores que terminem com a palavra “Maca”

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR LIKE '%Maca';
```

Estou fazendo uma consulta que retorne todos os produtos de sabores que iniciem com a palavra Morango

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE SABOR LIKE 'Morango%';
```

Estou fazendo uma consulta que retorne todos os produtos de sabores que iniciem com a palavra Morango e que sejam do tamanho PET

```
SELECT * FROM TABELA_DE_PRODUTOS
WHERE (SABOR LIKE 'Morango%') AND (EMBALAGEM = 'PET');
```

Consulta que diga quantos clientes possuem o sobrenome Silva.
Para mim não importa o nome que vem antes de 'Silva' e nem o nome que vem depois de 'Silva'

```
select * from TABELA_DE_CLIENTES where  NOME LIKE '%Silva%';
```

USANDO O DISTINCT

O SELECT DISTINCT instrução é usada para retornar apenas valores distintos (diferentes).

Dentro de uma tabela, uma coluna geralmente contém muitos valores duplicados e às vezes você deseja apenas listar os valores diferentes (distintos).

Trata-se de uma cláusula para eliminar repetições em consultas

```
USE SUCOS_FRUTAS;
```

```
SELECT * FROM TABELA_DE_PRODUTOS;
```

Essa consulta vai retornar o campo embalagem da tabela de produtos

```
SELECT EMBALAGEM FROM TABELA_DE_PRODUTOS;
```

Saber quantas embalagens diferentes tem meus produtos

```
SELECT DISTINCT EMBALAGEM FROM TABELA_DE_PRODUTOS;
```

Vai retornar todos os tipos de embalagens que o produto de sabor maça tem

```
SELECT DISTINCT EMBALAGEM FROM TABELA_DE_PRODUTOS WHERE
SABOR = 'Maca';
```

```
SELECT DISTINCT EMBALAGEM, SABOR FROM TABELA_DE_PRODUTOS;
```

Executando uma consulta para obter o número de sabores existentes

```
SELECT DISTINCT SABOR FROM TABELA_DE_PRODUTOS;
```

LIMITANDO A SAIDA DA CONSULTA

O comando TOP mostra os primeiros registros de uma tabela.

O comando TOP limita, mostrando as primeiras linhas daquela seleção que você está aplicando.

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_CLIENTES;
```

Dessa maneira vai mostrar na tela os 4 primeiros clientes

```
SELECT TOP 4 * FROM TABELA_DE_CLIENTES;

SELECT * FROM TABELA_DE_PRODUTOS;
```

Quero que mostre na tela somente os 5 primeiros produtos que estão cadastrados na tabela de produtos

```
SELECT TOP 5 * FROM TABELA_DE_PRODUTOS;
```

Quero que mostre na tela somente os 2 primeiros produtos que estão cadastrados na tabela de produtos e que sejam do sabor Maca

```
SELECT TOP 2 * FROM TABELA_DE_PRODUTOS WHERE SABOR = 'Maca';

SELECT * FROM NOTAS_FISCAIS;
```

Listando as 10 primeiras vendas do dia 01/10/2017

```
SELECT TOP 10 * FROM NOTAS_FISCAIS WHERE DATA_VENDA = '2017-10-01';
```

ORDENANDO A SAIDA

ASC significa ASCENDENTE - vai vir do menor para o maior
Quando eu estiver ordenando textos, menor para maior seria no sentido alfabético das letras, começando no A, terminando no Z.
Por padrão quando não colocamos "ASC" ou "DESC", significa que vai ser ASC
DESC significa DESCENDENTE - vai vir do maior para o menor
Quando eu estiver ordenando textos, maior para o menor começa no Z e termina no A

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_PRODUTOS;
```

Estou fazendo uma consulta que vai ser ordenada pelo preço do produto, como não coloquei "ASC" ou "DESC", por padrão a consulta vai ser ASC, ou seja, do menor preço para o maior preço

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY PRECO_DE_LISTA;
```

Estou fazendo uma consulta que vai ser ordenada pelo preço do produto, como coloquei "ASC" a consulta vai ser do menor preço para o maior preço

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY PRECO_DE_LISTA ASC;
```

Estou fazendo uma consulta que vai ser ordenada pelo preço do produto, como coloquei "DESC" a consulta vai ser do produto de maior preço para o menor preço

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY PRECO_DE_LISTA DESC;
```

Estou fazendo uma consulta que vai ser ordenada pelo nome do produto de forma descendente "DESC", e como o campo que vou ordenar e do tipo texto vai ser do Z até o A

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY NOME_DO_PRODUTO DESC;
```

Estou fazendo uma consulta que vai ser ordenada pelo nome do produto de forma ascendente "ASC", e como o campo que vou ordenar e do tipo texto vai ser do A até o Z, ou seja, no sentido alfabético das letras

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY NOME_DO_PRODUTO ;
```

Essa consulta terá o mesmo resultado da consulta de cima, porque mesmo que não colocamos ASC, por padrão a consulta sempre será do maior para o menor ou do sentido alfabético das letras

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY NOME_DO_PRODUTO ASC;
```

Essa consulta vai ser da seguinte maneira.
Estou ordenando primeiro por embalagem de forma Ascendente, nesse caso vai retornar primeiro os produtos que tiverem embalagem do tipo "Garrafa" em seguida do tipo "Lata" e por ultimo "PET" porque estou ordenando no sentido alfabetico das letras.
Dentro dos produtos que tem as embalagens do tipo Garrafa vou ordenar o nome deles de forma alfabetica ou seja do A ao Z

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY EMBALAGEM, NOME_DO_PRODUTO;
```

Estou ordenando tanto as embalagens dos produtos do Z ate A, quanto o nome do produto do Z ao A
Para ver o resultado inverso da consulta acima

```
SELECT * FROM TABELA_DE_PRODUTOS ORDER BY EMBALAGEM
DESC, NOME_DO_PRODUTO DESC;
```

Quero fazer uma consulta para saber os 5 produtos mais caros dentro dos produtos oferecidos pela empresa de suco de frutas.

```
SELECT TOP 5 * FROM TABELA_DE_PRODUTOS ORDER BY
PRECO_DE_LISTA DESC;
```

Quero fazer uma consulta para saber os 5 produtos mais baratos dentro dos produtos oferecidos pela empresa de suco de frutas.

```
SELECT TOP 5 * FROM TABELA_DE_PRODUTOS ORDER BY
PRECO_DE_LISTA ASC;
```

Desafio

Qual foi a maior venda do produto "Linha Refrescante - 1 Litro - Morango/Limao" em quantidade?

1º Passo: Buscar o código do produto "Linha Refrescante - 1 Litro - Morango/Limão":

```
SELECT * FROM TABELA_DE_PRODUTOS WHERE NOME_DO_PRODUTO
= 'Linha Refrescante - 1 Litro - Morango/Limao';
```

2° Passo: Com a consulta acima obtemos o resultado do código deste produto: '1101035'. Com esta informação em mãos, fazemos a consulta para achar a maior quantidade vendida deste produto:

```
SELECT * FROM ITENS_NOTAS_FISCAIS
WHERE CODIGO_DO_PRODUTO = 1101035 ORDER BY QUANTIDADE
DESC;
```

Constatou que a maior quantidade foi 99

3° Passo: Quantas vendas foram feitas com quantidade igual a 99 litros para o produto '1101035'?

```
SELECT COUNT(*) FROM ITENS_NOTAS_FISCAIS WHERE
CODIGO_DO_PRODUTO = '1101035' AND QUANTIDADE = 99;
```

O total foi de 79.
Portanto, essa foi a quantidade de vendas feitas para o produto 1101035.

```
SELECT CODIGO_DO_PRODUTO FROM ITENS_NOTAS_FISCAIS;
```

Verifique as quantidades totais de vendas de cada produto e ordene do maior para o menor.

Para obter o resultado esperado, nós podemos realizar uma consulta na tabela de itens notas fiscais, utilizando a função de agregação SUM para somar a quantidade e ordenar a saída utilizando o GROUP BY:

```
SELECT CODIGO_DO_PRODUTO, SUM(QUANTIDADE) AS
SOMA_DAS_QUANTIDADES FROM ITENS_NOTAS_FISCAIS
 GROUP BY CODIGO_DO_PRODUTO
 ORDER BY SUM(QUANTIDADE);
```

Agora, liste somente os produtos que venderam mais que 394000 unidades.

```
SELECT CODIGO_DO_PRODUTO, SUM(QUANTIDADE) AS
SOMA_DAS_QUANTIDADES FROM ITENS_NOTAS_FISCAIS
GROUP BY CODIGO_DO_PRODUTO
HAVING SUM(QUANTIDADE) > 394000
ORDER BY SUM(QUANTIDADE) DESC;
```

AGRUPANDO LINHAS DA TABELA

GROUP BY - agrupar os resultados da saída da consulta.
SUM - Soma
AVG - Média
MAX - Máximo
MIN - Mínimo

```
USE SUCOS_FRUTAS

SELECT * FROM TABELA_DE_CLIENTES;

SELECT CIDADE, IDADE FROM TABELA_DE_CLIENTES ORDER BY
CIDADE, IDADE;
```

Dentro do GROUP BY, o campo que está sendo somado não entra, só vai entrar aqui depois do group by os campos que são critérios de junção, de soma das linhas.

Nessa consulta abaixo quero saber a soma das idades dos clientes que moram em cada cidade

```
SELECT CIDADE, SUM(IDADE) AS SOMA_DAS_IDADE FROM
TABELA_DE_CLIENTES GROUP BY CIDADE;
```

Nessa consulta abaixo quero saber a soma das idades e a soma dos limites de credito dos clientes que moram em cada cidade

```
SELECT CIDADE, SUM(IDADE) AS SOMA_IDADE,
SUM(LIMITE_DE_CREDITO)  AS SOMA_DO_LIMITE_DE_CREDITO
FROM TABELA_DE_CLIENTES GROUP BY CIDADE;
```

Quero ver a média das idades dos clientes de cada cidade e a soma dos limites de créditos dos clientes de cada estado
Nesse caso a média de idade é 21 anos no Rio de Janeiro
e a média de idade e 27 em São Paulo

```
SELECT CIDADE, AVG(IDADE) AS MEDIA_DAS_IDADES,
SUM(LIMITE_DE_CREDITO) AS SOMA_DOS_LIMITES_DE_CREDITO
FROM TABELA_DE_CLIENTES GROUP BY CIDADE;
```

Contador de Linhas - Permite eu contar o número de clientes
que existem em uma cidade

```
SELECT CIDADE, COUNT(*) AS NUMERO_DE_CLIENTES FROM
TABELA_DE_CLIENTES GROUP BY CIDADE;
```

Estou contando o número de produtos por embalagem, somente do sabor laranja

```
SELECT EMBALAGEM, COUNT(*) AS NUMERO_DE_PRODUTOS FROM
TABELA_DE_PRODUTOS
WHERE SABOR = 'Laranja' GROUP BY EMBALAGEM;
```

HAVING para filtrar campos agregados

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_CLIENTES;
```

Quero listar os estado e saber a soma do limite de credito de todos os clientes de cada estado

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS
SOMA_DO_LIMITE_DE_CREDITO
FROM TABELA_DE_CLIENTES GROUP BY ESTADO;
```

Quero listar os estados cuja a soma do limite de credito e maior que 900 000

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS
SOMA_LIMITE_DE_CREDITO
FROM TABELA_DE_CLIENTES
GROUP BY ESTADO
HAVING SUM(LIMITE_DE_CREDITO) >= 900000;
```

Na tabela de produtos quero ver o maior preço e o menor preço de todos os produtos por tipo de embalagem

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM;
```

```
SELECT * FROM TABELA_DE_PRODUTOS;
```

Na tabela de produtos quero ver o maior preco e o menor preco dos produtos por tipo de embalagem e que o preço seja maior/igual a 10

```
SELECT EMBALAGEM,  MAX(PRECO_DE_LISTA) AS MAIOR_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA >= 10 GROUP
BY EMBALAGEM;
```

Na tabela de produtos quero ver o maior preco e o menor preco dos produtos por tipo de embalagem e que o preço seja menor/igual a 5

```
SELECT EMBALAGEM,  MAX(PRECO_DE_LISTA) AS MAIOR_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA <= 5 GROUP
BY EMBALAGEM;
```

Nesta consulta somente quero os produtos que possuem o maior preço de lista mais do que 20 unidade monetária,

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAXIMO_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA >= 10
GROUP BY EMBALAGEM HAVING MAX(PRECO_DE_LISTA) >=20;
```

CLASSIFICANDO OS CAMPOS

ESTRUTURA DE CLASSIFICAÇÃO

```
CASE WHEN <CONDIÇÃO> THEN <VALOR>
        WHEN <CONDIÇÃO> THEN <VALOR>
        WHEN <CONDIÇÃO> THEN <VALOR>
        ELSE <VALOR> END
```

Eu coloco CASE WHEN, uma condição lógica, que eu posso usar as mesmas coisas que eu utilizo lá no where, então eu posso colocar várias condições lógicas entre and, entre or, not, maior, menor, igual e assim por diante

O else é: se nenhuma condição for satisfeita, esse vai ser o valor final.

WHEN = QUANDO
THEN = ENTAO
ELSE = SENÃO

```
USE SUCOS_FRUTAS;
```

Quero que mostre na tela somente os produtos que sejam do sabor de Manga

```
SELECT NOME_DO_PRODUTO, PRECO_DE_LISTA FROM
TABELA_DE_PRODUTOS
WHERE SABOR = 'Manga';
```

Quero que mostre na tela somente os produtos que sejam do sabor de Manga
Quando o preço do produto for maior/igual a 12 então será considerado um produto caro
Quando o preço do produto for maior/igual a 7 e menor que 12 então será considerado um produto em conta
Se o produto não for "PRODUTO CARO" ou " PRODUTO EM CONTA" ele será um "PRODUTO BARATO”

```
SELECT NOME_DO_PRODUTO, PRECO_DE_LISTA,
(CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO'
        WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA'
        ELSE 'PRODUTO BARATO' END) AS CLASSIFICACAO
FROM TABELA_DE_PRODUTOS
WHERE SABOR = 'Manga'
ORDER BY CLASSIFICACAO;
```

Quero que mostre na tela todos os produtos e quando o preço do produto for maior/igual a 12 então será considerado um produto caro
Quando o preço do produto for maior/igual a 7 e menor que 12 então será considerado um produto em conta
Se o produto não for "PRODUTO CARO" ou " PRODUTO EM CONTA" ele será um "PRODUTO BARATO"

```
SELECT NOME_DO_PRODUTO, PRECO_DE_LISTA,
(CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO'
        WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA'
         ELSE 'PRODUTO BARATO' END) AS CLASSIFICACAO
FROM TABELA_DE_PRODUTOS
ORDER BY CLASSIFICACAO;  -- aqui está sendo ordenado de forma alfabética, estou utilizando o alias
```

Estou contando quantos produtos baratos, caros e em conta eu tenho

```
SELECT
(CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO'
        WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA'
         ELSE 'PRODUTO BARATO' END) AS CLASSIFICACAO, COUNT(*) AS NUMERO_DE_PRODUTOS
FROM TABELA_DE_PRODUTOS
GROUP BY (CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO'
        WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA'
    ELSE 'PRODUTO BARATO' END);
```

O GROUP BY não aceita Alias

DESAFIO

Para cada cliente temos seus limites de crédito mensais. Liste somente o nome dos clientes e os classifique por:

- Acima ou igual a 150.000 - Clientes grandes
- Entre 150.000 e 110.000 - Clientes médios
- Menores que 110.000 - Clientes pequenos

```
SELECT * FROM TABELA_DE_CLIENTES;
```

```
SELECT CPF, NOME, LIMITE_DE_CREDITO,
(CASE WHEN LIMITE_DE_CREDITO >= 150000 THEN 'CLIENTE GRANDE'
      WHEN LIMITE_DE_CREDITO >= 110000 AND LIMITE_DE_CREDITO < 150000 THEN 'CLIENTE MÉDIO'
      ELSE 'CLIENTE PEQUENO' END) AS CLASSIFICACAO_DO_CLIENTE
FROM TABELA_DE_CLIENTES
ORDER BY LIMITE_DE_CREDITO DESC;
```

INNER JOIN

JOIN "**JUNÇÃO**"

Os INNER JOIN têm mais performance quando eu tenho índices nos campos que eu estou juntando.
Quando eu tenho uma chave estrangeira, esse índice já é naturalmente criado pelo banco de dados.
Então INNER JOINS entre chaves estrangeiras têm mais performance, porém eu posso fazer INNER JOINS entre dois campos que não têm uma chave estrangeira, desde que sejam campos com o mesmo conteúdo e do mesmo tipo.

```
USE SUCOS_FRUTAS;
```

```
SELECT * FROM TABELA_DE_VENDEDORES;
```

```
SELECT * FROM NOTAS_FISCAIS;
```

Se eu estou usando **COUNT**, estou usando uma função de agrupamento então tem que colocar o **Group By**

Contando quantas matriculas temos, ou seja, contar quantas notas fiscais cada vendedor vendeu

```
SELECT MATRICULA, COUNT(*) AS NUMERO_DE_VENDAS
FROM NOTAS_FISCAIS GROUP BY MATRICULA;
```

Quero que mostre na tela o campo “matrícula dos vendedores” da tabela notas fiscais.
Quero que mostre na tela o campo “nome dos vendedores” da tabela de vendedores e quero que contabilize quantas vendas cada vendedor fez

```
SELECT NOTAS_FISCAIS.MATRICULA,
TABELA_DE_VENDEDORES.NOME, COUNT(*) AS NUMERO_DE_VENDAS
FROM NOTAS_FISCAIS
INNER JOIN TABELA_DE_VENDEDORES
ON NOTAS_FISCAIS.MATRICULA =
TABELA_DE_VENDEDORES.MATRICULA
GROUP BY NOTAS_FISCAIS.MATRICULA,
TABELA_DE_VENDEDORES.NOME;
```

Fazendo a mesma consulta acima, mas utilizando o Alias/Apelido
Na consulta acima estou utilizando o nome da tabela

```
SELECT NF.MATRICULA, TV.NOME, COUNT(*) AS NUMERO_DE_VENDAS
FROM NOTAS_FISCAIS AS NF
INNER JOIN TABELA_DE_VENDEDORES AS TV
ON NF.MATRICULA = TV.MATRICULA
GROUP BY NF.MATRICULA, TV.NOME;
```

Posso inverter as ordens da tabela

```
SELECT NF.MATRICULA, TV.NOME, COUNT(*) AS NUMERO_DE_VENDAS
FROM TABELA_DE_VENDEDORES  AS TV
INNER JOIN  NOTAS_FISCAIS AS NF
ON NF.MATRICULA = TV.MATRICULA
GROUP BY NF.MATRICULA, TV.NOME;
```

LEFT JOIN – Vai buscar todo mundo que está à
esquerda do JOIN e somente os em comuns na direita de JOIN

RIGHT JOIN - Vai pegar somente quem tem em comum na tabela da esquerda e todo mundo da tabela da direita.

FULL JOIN - Vai pegar todo mundo da esquerda e todo mundo da direita.

CROSS JOIN - Vai pegar a análise combinatória de todo mundo. Então ele vai pegar a primeira linha da tabela da esquerda e cruzar com as quatro da direita

LEFT JOIN

Consulta para verificar os clientes que tiveram vendas, nesse caso vai ter dados repetidos, porque um cliente pode ter feito várias compras

```
SELECT TC.CPF AS CPF_DO_CADASTRO, TC.NOME AS
NOME_DO_CLIENTE,
NF.CPF AS CPF_DA_TABELA_NOTA
FROM TABELA_DE_CLIENTES AS TC
INNER JOIN
NOTAS_FISCAIS AS NF
ON TC.CPF = NF.CPF;
```

Consulta que permite ver quantos clientes compraram na minha empresa, não vai vir dados repetidos

```
SELECT DISTINCT
TC.CPF AS CPF_DO_CADASTRO, TC.NOME AS NOME_DO_CLIENTE,
NF.CPF AS CPF_DA_TABELA_NOTA
FROM TABELA_DE_CLIENTES AS TC
INNER JOIN
NOTAS_FISCAIS AS NF
ON TC.CPF = NF.CPF;
```

Quando estou usando o **COUNT** estou contando o número de linhas da tabela
Quando estou usando o **COUNT** de forma isolada eu não preciso utilizar o **GROUP BY**, pois não tem nenhum campo para agrupar

```
SELECT COUNT(*) FROM TABELA_DE_CLIENTES;
```

Incluindo um cliente novo

```
INSERT INTO TABELA_DE_CLIENTES
(CPF, NOME, ENDERECO_1, ENDERECO_2, BAIRRO, CIDADE, ESTADO, CEP,
DATA_DE_NASCIMENTO,IDADE, GENERO, LIMITE_DE_CREDITO, VOLUME_DE_COMPRA,
PRIMEIRA_COMPRA)
VALUES ('23412632331', 'Juliana Silva', 'Rua Tramandaí', ' ', 'Bangu', 'Rio de
Janeiro', 'RJ', '23400000', '1989-02-04', 33, 'F', 180000, 24500, 0);
```

PRATICANDO O LEFT JOIN

Ver os clientes que pelo menos fizeram uma venda

A esquerda do "LEFT JOIN" e a tabela de clientes e a direita do "LEFT JOIN" e a tabela de notas fiscais, logo que estou usando o "LEFT JOIN" vou ver todo mundo que está a esquerda e só quem tem combinação na direita, ou seja, vou trazer todos os clientes da tabela de clientes e só os que compraram da tabela de notas fiscais

```
SELECT DISTINCT
TC.CPF AS CPF_DO_CADASTRO, TC.NOME AS NOME_DO_CLIENTE,
NF.CPF AS CPF_DA_TABELA_NOTA
FROM TABELA_DE_CLIENTES AS TC LEFT JOIN NOTAS_FISCAIS AS NF
ON TC.CPF = NF.CPF;
```

Qual foi o cliente que não fez nenhuma venda

```
SELECT DISTINCT
TC.CPF AS CPF_DO_CADASTRO,
TC.NOME AS NOME_DO_CLIENTE
FROM TABELA_DE_CLIENTES AS TC
LEFT JOIN
NOTAS_FISCAIS AS NF
ON TC.CPF = NF.CPF
WHERE NF.CPF IS NULL;
```

OUTROS JOIN

```
USE SUCOS_FRUTAS;
```

Contar quantos clientes eu tenho
Nesse caso tenho 16 clientes

```
SELECT COUNT(*) FROM TABELA_DE_CLIENTES;
```

Contar quantos vendedores eu tenho
Nesse caso tenho 4 vendedores

```
SELECT COUNT(*) FROM TABELA_DE_VENDEDORES;
```

Consulta que permite verificar quais os clientes e os vendedores que moram em determinados bairros

```
SELECT DISTINCT
TV.NOME AS NOME_DO_VENDEDOR,
TV.BAIRRO AS BAIRRO_DO_VENDEDOR,
TC.NOME AS NOME_DO_CLIENTE,
TC.BAIRRO AS BAIRRO_DO_CLIENTE
```

```
FROM
TABELA_DE_CLIENTES AS TC
INNER JOIN
TABELA_DE_VENDEDORES AS TV
ON TC.BAIRRO = TV.BAIRRO;
```

Consulta que permite verificar qual é o vendedor que não tem cliente no seu bairro

```
SELECT DISTINCT
TV.NOME AS NOME_DO_VENDEDOR,
TV.BAIRRO AS BAIRRO_DO_VENDEDOR,
TC.NOME AS NOME_DO_CLIENTE,
TC.BAIRRO AS BAIRRO_DO_CLIENTE
FROM TABELA_DE_CLIENTES AS TC RIGHT JOIN
TABELA_DE_VENDEDORES AS TV
ON TC.BAIRRO = TV.BAIRRO;
```

```
SELECT DISTINCT
TV.NOME AS NOME_DO_VENDEDOR,
TV.BAIRRO AS BAIRRO_DO_VENDEDOR,
TC.NOME AS NOME_DO_CLIENTE,
TC.BAIRRO AS BAIRRO_DO_CLIENTE
FROM TABELA_DE_CLIENTES AS TC RIGHT JOIN
TABELA_DE_VENDEDORES AS TV
ON TC.BAIRRO = TV.BAIRRO
WHERE TC.NOME IS NULL;
```

Explicação do resultado da consulta acima: Vendedor que tiver nome do cliente e bairro do cliente nulo e um cliente que n tem nenhum vendedor no seu bairro, ou seja, nenhum cliente mora no bairro de Copacabana que é o bairro que o vendedor mora

Qual é o cliente que n possuem vendedores no seu bairro
Quero trazer todos os clientes, mas somente alguns vendedores

```
SELECT DISTINCT
TV.NOME AS NOME_DO_VENDEDOR,
TV.BAIRRO AS BAIRRO_DO_VENDEDOR,
TC.NOME AS NOME_DO_CLIENTE,
TC.BAIRRO AS BAIRRO_DO_CLIENTE
FROM TABELA_DE_CLIENTES AS TC LEFT JOIN
TABELA_DE_VENDEDORES AS TV
ON TC.BAIRRO = TV.BAIRRO
WHERE TV.NOME IS NULL;
```

Essa consulta vai retornar todos os clientes que não estão em bairros que tenham vendedores

```
SELECT DISTINCT
TV.NOME AS NOME_DO_VENDEDOR,
```

```
TV.BAIRRO AS BAIRRO_DO_VENDEDOR,
TC.NOME AS NOME_DO_CLIENTE,
TC.BAIRRO AS BAIRRO_DO_CLIENTE
FROM TABELA_DE_CLIENTES AS TC FULL JOIN
TABELA_DE_VENDEDORES AS TV
ON TC.BAIRRO = TV.BAIRRO;
```

UNION

Union Union All

```
Use SUCOS_FRUTAS;
```

Consulta para verificar quantos bairros tem relação com os clientes

```
SELECT DISTINCT BAIRRO FROM TABELA_DE_CLIENTES;
```

Consulta para verificar quantos bairros tem relação com os vendedores

```
SELECT DISTINCT BAIRRO FROM TABELA_DE_VENDEDORES;
```

Union permite fazer consultas

```
SELECT DISTINCT BAIRRO FROM TABELA_DE_CLIENTES
UNION
SELECT DISTINCT BAIRRO FROM TABELA_DE_VENDEDORES;
```

Essa consulta e a mesma da de cima como se fosse um DISTINCT
Ele não vai mostrar nada repetido

```
SELECT BAIRRO FROM TABELA_DE_CLIENTES
UNION
SELECT BAIRRO FROM TABELA_DE_VENDEDORES;
```

Union All não aplica o distinct sobre o resultado,
O resultado final vai vim com dados repetidos

```
SELECT BAIRRO FROM TABELA_DE_CLIENTES
UNION ALL
SELECT BAIRRO FROM TABELA_DE_VENDEDORES;
```

Nesse caso estou dizendo a origem desse dado
O **'CLIENTE', 'FORNECEDOR'** e uma constante que vai aparecer em cada linha

```
SELECT DISTINCT BAIRRO, 'CLIENTE' AS ORIGEM FROM
TABELA_DE_CLIENTES
UNION ALL
SELECT DISTINCT BAIRRO, 'FORNECEDOR' AS ORIGEM FROM
TABELA_DE_VENDEDORES;
```

O 'CLIENTE', 'FORNECEDOR' é uma constante que vai aparecer em cada linha

```
SELECT DISTINCT BAIRRO, 'CLIENTE' AS ORIGEM FROM
TABELA_DE_CLIENTES
UNION
SELECT DISTINCT BAIRRO, 'FORNECEDOR' AS ORIGEM FROM
TABELA_DE_VENDEDORES;
```

HAVING

O **HAVING** é usado quando nós queremos usar o resultado de uma agregação no nosso filtro.

```
USE SUCOS_FRUTAS;

SELECT * FROM TABELA_DE_CLIENTES;
```

Quero listar os estados e saber a soma do limite de crédito de todos os clientes de cada estado

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS SOMA_DO_LIMITE_DE_CREDITO
FROM TABELA_DE_CLIENTES GROUP BY ESTADO;
```

Quero listar os estados cuja soma do limite de credito e maior que 900000

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS SOMA_LIMITE_DE_CREDITO
FROM TABELA_DE_CLIENTES
GROUP BY ESTADO
HAVING SUM(LIMITE_DE_CREDITO) >= 900000;
```

Na tabela de produtos quero ver o maior preço e o menor preço de todos os produtos por tipo de embalagem

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM;

SELECT * FROM TABELA_DE_PRODUTOS;
```

Na tabela de produtos quero ver o maior preço e o menor preço dos produtos por tipo de embalagem e que o preço seja maior/igual a 10

```
SELECT EMBALAGEM,  MAX(PRECO_DE_LISTA) AS MAIOR_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA >= 10 GROUP BY
EMBALAGEM;
```

Na tabela de produtos quero ver o maior preço e o menor preço dos produtos por tipo de embalagem e que o preço seja menor/igual a 5

```
SELECT EMBALAGEM,  MAX(PRECO_DE_LISTA) AS MAIOR_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA <= 5 GROUP
BY EMBALAGEM;
```

Nesta consulta somente quero os produtos que possuem o maior preço de lista mais do que 20 unidade monetária,

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAXIMO_PRECO ,
MIN(PRECO_DE_LISTA) AS MENOR_PRECO
FROM TABELA_DE_PRODUTOS WHERE PRECO_DE_LISTA >= 10
GROUP BY EMBALAGEM HAVING MAX(PRECO_DE_LISTA) >=20;
```

DESAFIO: NOME DOS PRODUTOS

Em exercícios anteriores, pretendíamos obter os produtos que venderam mais que 394000 litros

```
SELECT CODIGO_DO_PRODUTO, SUM(QUANTIDADE) AS QUANTIDADE FROM
ITENS_NOTAS_FISCAIS
GROUP BY CODIGO_DO_PRODUTO HAVING SUM(QUANTIDADE) > 394000
ORDER BY SUM(QUANTIDADE) DESC;
```

Dito isso, e levando em consideração os comandos da consulta, desejamos que na resposta desta consulta apareça não somente o código do produto, mas também o nome do produto.

```
SELECT ITENS_NOTAS_FISCAIS.CODIGO_DO_PRODUTO, TABELA_DE_PRODUTOS.NOME_DO_PRODUTO,
SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE) AS QUANTIDADE
FROM ITENS_NOTAS_FISCAIS
INNER JOIN TABELA_DE_PRODUTOS
ON TABELA_DE_PRODUTOS.CODIGO_DO_PRODUTO = ITENS_NOTAS_FISCAIS.CODIGO_DO_PRODUTO
GROUP BY ITENS_NOTAS_FISCAIS.CODIGO_DO_PRODUTO, TABELA_DE_PRODUTOS.NOME_DO_PRODUTO
HAVING SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE) > 394000
ORDER BY SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE) DESC;
```

```
SELECT ITENS_NOTAS_FISCAIS.CODIGO_DO_PRODUTO, TABELA_DE_PRODUTOS.NOME_DO_PRODUTO
FROM TABELA_DE_PRODUTOS
INNER JOIN ITENS_NOTAS_FISCAIS
ON TABELA_DE_PRODUTOS.CODIGO_DO_PRODUTO = ITENS_NOTAS_FISCAIS.CODIGO_DO_PRODUTO;
```

Subconsultas no comando IN

Sub Query pode ser usada como se fosse uma tabela
Exemplo:

```
SELECT SUBTABELA. * FROM
(
        SELECT CLIENTE, CIDADE, ESTADO
        FROM CLIENTE
        UNION
        SELECT FORNECEDOR, CIDADE, ESTADO
        FROM FORNECEDOR
)
SUBTABELA
```

Neste caso é obrigatório usar um ALIAS para a SubQuery, nesse caso o alias é "SUBTABELA"

```
USE SUCOS_FRUTAS;
```

Consulta que vai retornar todos os bairros onde os meus vendedores possuem escritórios, não vai repetir bairro

```
SELECT DISTINCT BAIRRO FROM TABELA_DE_VENDEDORES;
```

Listando os bairros dos clientes onde eu tenho vendedores

```
SELECT BAIRRO FROM TABELA_DE_CLIENTES
WHERE BAIRRO IN ('Copacabana', 'Jardins', 'Santo Amaro', 'Tijuca');
```

PARA UTILIZAR UMA SUBQUERY DENTRO DO "IN" E OBRIGATORIO QUE O RESULTADO DA SUBQUERY TENHA SOMENTE UM CAMPO
EXEMPLO:
A SUBQUERY SELECT DISTINCT BAIRRO FROM TABELA_DE_VENDEDORES TER SOMENTE O CAMPO "BAIRRO"

```
SELECT BAIRRO FROM TABELA_DE_CLIENTES
WHERE BAIRRO IN (SELECT DISTINCT BAIRRO FROM
TABELA_DE_VENDEDORES);
```

SUBCONSULTAS SUBSTITUINDO O HAVING

Quando usamos sub query dentro de um from, eu sou obrigado a colocar alias.

Média do preço de lista por embalagens

```
SELECT EMBALAGEM, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM;
```

Quero saber as embalagens que são menores do que o preço médio igual a 10

```
SELECT EMBALAGEM, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM
HAVING AVG(PRECO_DE_LISTA) <= 10;
```

Quero saber as embalagens que são menores do que o preço médio igual a 10
CONSULTA UTILIZANDO SUBQUERY

```
SELECT MEDIA_EMBALAGENS.EMBALAGEM,
MEDIA_EMBALAGENS.PRECO_MEDIO FROM
(SELECT EMBALAGEM, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM) AS MEDIA_EMBALAGENS
WHERE MEDIA_EMBALAGENS.PRECO_MEDIO <= 10;
```

DESAFIO TRANSFORMANDO HAVING EM SUBCONSULTAS

Antes de qualquer coisa, vamos separar a consulta que nos dá o valor das quantidades agrupadas.

```
SELECT INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO, SUM(INF.QUANTIDADE) AS
QUANTIDADE FROM ITENS_NOTAS_FISCAIS  INF
INNER JOIN TABELA_DE_PRODUTOS TP
ON INF.CODIGO_DO_PRODUTO = TP.CODIGO_DO_PRODUTO
GROUP BY INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO;
```

Observação importante: Não se esqueça de incluir um apelido para SUM(QUANTIDADE) e outro para a consulta que ficará dentro do FROM.

```
SELECT SC.CODIGO_DO_PRODUTO, SC.NOME_DO_PRODUTO, SC.QUANTIDADE_TOTAL
FROM
(SELECT INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO, SUM(INF.QUANTIDADE) AS
QUANTIDADE_TOTAL FROM ITENS_NOTAS_FISCAIS  INF
INNER JOIN TABELA_DE_PRODUTOS TP
ON INF.CODIGO_DO_PRODUTO = TP.CODIGO_DO_PRODUTO
GROUP BY INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO) SC;
```

Finalmente, portanto, aplicamos a ordenação e o filtro.

```
SELECT SC.CODIGO_DO_PRODUTO, SC.NOME_DO_PRODUTO, SC.QUANTIDADE_TOTAL
FROM
(SELECT INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO, SUM(INF.QUANTIDADE) AS
QUANTIDADE_TOTAL FROM ITENS_NOTAS_FISCAIS  INF
INNER JOIN TABELA_DE_PRODUTOS TP
ON INF.CODIGO_DO_PRODUTO = TP.CODIGO_DO_PRODUTO
GROUP BY INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO) SC
WHERE SC.QUANTIDADE_TOTAL > 394000
ORDER BY SC.QUANTIDADE_TOTAL DESC;
```

VISÃO - VIEW

Criando uma View. Recomendado deixar essa view na primeira linha do seu Script

```
CREATE VIEW MEDIA_EMBALAGENS AS
SELECT EMBALAGEM, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM
```

```
SELECT * FROM MEDIA_EMBALAGENS;
```

Quero saber as embalagens que são menores do que o preço médio igual a 10
CONSULTA UTILIZANDO SUBQUERY

```
SELECT MEDIA_EMBALAGENS.EMBALAGEM,
MEDIA_EMBALAGENS.PRECO_MEDIO FROM
(SELECT EMBALAGEM, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO
FROM TABELA_DE_PRODUTOS GROUP BY EMBALAGEM) AS
MEDIA_EMBALAGENS
WHERE MEDIA_EMBALAGENS.PRECO_MEDIO <= 10;
```

Quero saber as embalagens que são menores do que o preço médio igual a 10
CONSULTA UTILIZANDO VISÃO/VIEW, e a mesma consulta acima, só muda que utilizo view

```
SELECT  EMBALAGEM, PRECO_MEDIO
FROM MEDIA_EMBALAGENS
WHERE PRECO_MEDIO <= 10;
```

DESAFIO:USANDO VISÃO PARA SUBSTITUIR O HAVING

Redesenhe esta consulta criando uma visão para a lista de quantidades totais por produto e aplicando a condição e ordenação sobre esta mesma visão.

```
SELECT INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO, SUM(INF.QUANTIDADE) AS QUANTIDADE FROM
ITENS_NOTAS_FISCAIS  INF
INNER JOIN TABELA_DE_PRODUTOS TP
ON INF.CODIGO_DO_PRODUTO = TP.CODIGO_DO_PRODUTO
GROUP BY INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO HAVING SUM(INF.QUANTIDADE) > 394000
ORDER BY SUM(INF.QUANTIDADE) DESC;
```

1° Passo: Vamos criar a visão com a consulta que retorna as quantidades agregadas. Não se esqueça de criar um apelido para o agregador SUM(QUANTIDADE).

```
CREATE VIEW VW_QUANTIDADE_PRODUTOS AS SELECT INF.CODIGO_DO_PRODUTO,
TP.NOME_DO_PRODUTO,
SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL FROM ITENS_NOTAS_FISCAIS  INF
INNER JOIN TABELA_DE_PRODUTOS TP
ON INF.CODIGO_DO_PRODUTO = TP.CODIGO_DO_PRODUTO
GROUP BY INF.CODIGO_DO_PRODUTO, TP.NOME_DO_PRODUTO;
```

2° Passo: Consulta redesenhada, utilizando a visão

```
SELECT * FROM VW_QUANTIDADE_PRODUTOS
WHERE QUANTIDADE_TOTAL > 394000
ORDER BY QUANTIDADE_TOTAL DESC;
```

FUNÇÕES NO SQL

Tipos de Funções

Strings (Textos)
Datas

Matematicas
Conversão de dados

Funções no SQL do tipo Texto

A função **"LOWER"** pega uma string ou uma expressão de caracteres e vai converter essa expressão de caracteres apenas para caracteres minúsculos. Para usar a função, coloco '**LOWER**' e entre aspas simples o campo ou o texto que eu quero aplicar a função.

```
SELECT NOME, LOWER(NOME) AS NOME_MINUSCULO
FROM TABELA_DE_CLIENTES;
```

A próxima função é a **"UPPER",** que é o inverso do LOWER. Essa função vai pegar uma expressão de caracteres e vai converter tudo que é minúsculo em maiúsculo.

```
SELECT NOME, LOWER(NOME) AS NOME_MINUSCULO, UPPER(NOME)
AS NOME_MAIUSCULO FROM TABELA_DE_CLIENTES;
```

```
SELECT NOME,UPPER(NOME) AS NOME_MAIUSCULO FROM
TABELA_DE_CLIENTES;
```

A função **"CONCAT"** vai retornar uma cadeia de caracteres ou uma string, resultante da concatenação, ou seja, da junção de dois ou mais outros valores de caracteres.
Uma observação: o SQL Server também aceita que eu pegue essas strings e coloque entre eles o símbolo de somar.

```
SELECT NOME, CONCAT(ENDERECO_1, ' ', BAIRRO, ' ',CIDADE, ' ', ESTADO, ' - ', CEP) AS
ENDERECO_COMPLETO
FROM TABELA_DE_CLIENTES;
```

Outra maneira de fazer a mesma consulta acima

```
SELECT NOME, ENDERECO_1 + ' ' + BAIRRO + ' ' + CIDADE + ' ' + ESTADO + ' - ' + CEP AS
ENDERECO_COMPLETO
FROM TABELA_DE_CLIENTES;
```

A função **"RIGHT"** vai retornar a parte direita de uma cadeia de caracteres, usando como parâmetro o número de caracteres especificados na segunda parte da função.
A função **"LEFT"** faz o inverso do RIGHT, retornando a parte esquerda de uma cadeia de caracteres, sempre usando o número de caracteres especificados no segundo parâmetro da função.

Consulta que vai retornar os três primeiros caracteres

```
SELECT NOME_DO_PRODUTO, LEFT(NOME_DO_PRODUTO,3) AS
TRES_PRIMEIROS_CHAR
FROM TABELA_DE_PRODUTOS;
```

A função **“REPLICATE”** vai repetir o valor de caracteres que está na função, usando como parâmetro o número especificado como número de vezes.
O **“REPLACE”** vai substituir um conjunto de caracteres por outro especificado.

Onde tiver Litros será substituído por L

```
SELECT TAMANHO, REPLACE(TAMANHO, 'Litros', 'L') AS TAMANHO_MODIFICADO
FROM TABELA_DE_PRODUTOS;
```

Utilizando função dentro de uma função
Onde tiver Litros será substituído por L
Onde tiver Litro "no singular" também será substituído por L

```
SELECT TAMANHO, REPLACE((REPLACE(TAMANHO, 'Litros', 'L')), 'Litro', 'L') AS
TAMANHO_MODIFICADO FROM TABELA_DE_PRODUTOS;
```

A função **“SUBSTRING”** vai retornar uma parte da expressão de caracteres, partindo do ponto inicial que será passado como parâmetro para a função e o número de caracteres a serem extraídos.

A função **“LTRIM”** remove os caracteres que são espaços

A função **“RTRIM”** faz a mesma coisa que o **LTRIM**, porém do lado direito.

A função “**TRIM”** remove tanto os espaços da esquerda quanto os espaços da direita:

O **“REPLACE”** vai substituir um conjunto de caracteres por outro especificado.

O **“LEN”** retorna o número de caracteres do texto que está sendo especificado, excluindo os espaços que possa haver à direita.

DESAFIO - BUSCANDO UM PEDAÇO DE UM TEXTO

```
USE SUCOS_FRUTAS;
```

Conforme vimos nas aulas sobre funções de texto, observe a seguinte frase abaixo:
CIDADE DO RIO DE JANEIRO

Como seria a função para retirar deste texto somente a palavra RIO?

Resposta:

```
SUBSTRING('CIDADE DO RIO DE JANEIRO', 11, 3)
```

Desafio: separando nome e sobrenome

```
SELECT * FROM TABELA_DE_CLIENTES;
```

Nas aulas relacionadas às funções de texto, observamos a tabela de cliente onde podemos listar os nomes.

Note que os nomes e sobrenomes são separados por um espaço.

Sendo assim, faça uma consulta que traga somente o primeiro nome de cada cliente.

Dica: Como foi dito pelo instrutor existem diversas funções do SQL Server e muitas vezes não sabemos todas. Por isso, para resolver este problema pesquise sobre a função "**CHARINDEX"**, veja como ela funciona, e aplique para resolver este problema.

1° PASSO: Primeiro, vamos localizar o primeiro espaço do nome. Sua posição.
Para isso, basta executar a função "CHARINDEX"
Isso ' ' que dizer que estou procurando onde tem espaço
Estou procurando na coluna "NOME", a pesquisa vai iniciar da posição "1"

```
SELECT NOME, CHARINDEX(' ',NOME, 1) FROM TABELA_DE_CLIENTES;
```

2° PASSO: Usando a função **"SUBSTRING"** podemos buscar parte do texto que compõe o nome completo buscando da posição 1 a posição do primeiro espaço.

```
SELECT NOME, SUBSTRING(NOME, 1, CHARINDEX(' ', NOME, 1)) FROM
TABELA_DE_CLIENTES;
```

```
SELECT NOME,SUBSTRING(NOME, 1, CHARINDEX(' ',NOME)) AS
PRIMEIRO_NOME FROM TABELA_DE_CLIENTES;
```

3° Passo: Quero que retorne, o primeiro e o segundo nome

```
SELECT NOME, LEFT(NOME, CHARINDEX(' ', NOME)) AS [PRIMEIRO
NOME],
RIGHT(NOME, LEN(NOME) - CHARINDEX(' ', NOME)) AS [SOBRENOME]
FROM TABELA_DE_CLIENTES;
```

Utilizei a função **"LEFT"** para retornar o primeiro nome em conjunto com a função **"CHARINDEX"**.
De quebra ainda adicionei na query o último nome também.
O número 20 indica o tamanho da string retornada da função da substring.
Então, a função charindex localiza o primeiro espaço dentro da coluna nome e com a função substring eu começo a contar a quantidade de caracteres depois da localização até no máximo 20. Dessa forma não fica limitado somente o retorno da quantidade de caracteres contados pela função charindex.

```
SELECT CPF, LEFT(NOME,CHARINDEX(' ',NOME)) AS PRIMEIRO_NOME,
SUBSTRING(NOME, CHARINDEX(' ',NOME), 20 ) AS ULTIMO_NOME,
NOME AS NOME_COMPLETO
FROM TABELA_DE_CLIENTES;
```

FUNÇÕES DE DATA E HORA

DATEADD - Essa função adiciona um número (um inteiro com sinal positivo ou negativo) a um datepart de uma data de entrada e retorna um valor de data/hora modificado

DATEPART é uma parte da data, o datepart pode ser a palavra year, quarter, month, dayofyear, day, week, hour, minute, second, milisecond, microsecond, nanosecond

DATEPART pode ser usada sobre as funções **"dateadd"** e **"datediff"**

EXEMPLO:

Eu escrevo dateadd e escolho um datepart que nesse caso e o "day".
Então se eu escrever day, passar como parâmetro um número inteiro positivo ou negativo, no caso eu escolhi um positivo, o valor 30, e uma data, ele vai somar 30 dias àquela data:

DATEADD(DAY, 30, '2022-01-01')

Então, o resultado será aquela mesma data 30 dias para frente:

2022-01-31

Se eu tivesse escolhido o datepart month, ele irá somar 30 meses à data, então ele vai sempre somar ou diminuir, dependendo se o valor passado como parâmetro for positivo ou negativo.

DATEDIFF - Essa função retorna a contagem (como um valor inteiro com sinal) dos limites de datepart especificados cruzados entre os parâmetros especificados startdate e enddate

Função datediff vai retornar um número, que é um valor inteiro, que pode ser positivo ou negativo, que expressa os dateparts da diferença entre duas datas.

EXEMPLO:

DATA INICIAL '2022-01-01'
DATA FINAL '2022-04-12'
DATEDIFF(DAY, '2022-01-01', '2022-04-12')

A função DATEDIFF calculará a diferença entre essas duas datas e mostrará o resultado expresso em dias, porque o datepart escolhido foi day.
Se eu tivesse escrito month, o resultado da diferença entre essas duas datas seria expresso em meses, e assim por diante.
Logo, o resultado do exemplo seria 101 dias.

DATEPART - Essa função retorna um inteiro que representa o datepart especificado do argumento date especificado.

Com a função datepart, passamos uma data e posso ver o ano, o mês, o dia, a hora, o minuto dessa data.
Então se eu escrevo:

DATEPART(DAY, '2022-01-01')

Nos será retornado o dia dessa data, logo o resultado será 1.

GETDATE - Retorna o carimbo de data/hora do sistema do banco de dados atual como um valor de datetime sem o deslocamento de fuso horário do banco de dados
Esse valor é derivado do sistema operacional do computador no qual a instância do SQL Server está sendo executada

A função getdate vai retornar a data do computador de onde a instância do banco de dados está rodando. Então se eu executo a função GETDATE(), por exemplo, eu vou ter a data, a hora, o minuto, o segundo e o milissegundo do momento em que eu executei a função.

DAY - Esta função retorna um inteiro que representa o dia (dia do mês) da data especificada.
MONTH - Retorna um inteiro que representa o mês da data especificada
YEAR - Retorna um inteiro que representa o ano da data especificada

A função day, se eu colocar DAY(DATE), eu vou ver o dia da data. Semelhante à função datepart, quando eu uso day como parâmetro.

Na função month, eu vou ver o mês da data e na função year, verei o ano da data.

ISDATE - Retornará 1 se a expressão for um valor datetime, válido; caso contrário, 0.

A função isdate testa para saber se a expressão passada por parâmetro é uma data válida ou não. Então por exemplo, se eu escrevo:

ISDATE('2022-02-31')

A data 31 de fevereiro de 2022 não existe, então o resultado vai ser o número 0, de falso.
Se a data fosse válida, a função isdate retornaria o número 1.

DATETIMEFROMPARTS - Essa função retorna um valor datetime para os argumentos de data e hora especificados

A função datetimefromparts vai retornar uma data baseado em inteiros separados por vírgula, onde eu vou expressar o ano, o mês, o dia, a hora, o minuto, o segundo e o milissegundo:

DATETIMEFROMPARTS(year, month, day, hour, minute, seconds, milliseconds)

Então se eu escrevo:

DATETIMEFROMPARTS(2022, 12, 14, 15, 34, 22, 30)

Ou seja, eu quero que essa função retorne uma data onde o ano é 2022, o mês é 12, o dia é 14 e assim por diante:

2022-12-14 15:34:22.030

```
USE SUCOS_FRUTAS;
```

Executando essa consulta, o resultado será a data e hora do computador

```
SELECT GETDATE();
```

Vamos pegar o dia atual e somar dez dias:

```
SELECT DATEADD(DAY, 10, GETDATE());
```

Vamos pegar o dia atual e somar dez dias:

```
SELECT GETDATE() AS DATA_HOJE, DATEADD(DAY, 10, GETDATE()) AS DATA_DAQUI_10_DIAS;
```

Se eu quiser saber qual data representa 48 dias atrás, basta eu fazer:

```
SELECT DATEADD(DAY,-48, GETDATE()) AS DATA_48_DIAS_ATRAS
```

```
SELECT DATEADD(DAY,-48, GETDATE()) AS DATA_48_DIAS_ATRAS,
DATEDIFF(DAY, '2023-01-01', GETDATE()) AS DIAS_DESDE_INICIO_ANO;
```

Eu quero saber por exemplo quantos dias se passaram desde o primeiro dia do ano:

```
SELECT DATEDIFF(DAY, '2023-01-01', GETDATE()) AS
DIAS_DESDE_INICIO_ANO;
```

Se eu quiser ver o número de horas desde o primeiro dia do ano:

```
SELECT DATEDIFF(HOUR, '2023-01-01', GETDATE()) AS
HORAS_DESDE_INICIO_ANO;
```

Se eu quiser ver o número de meses desde o primeiro mês do ano:

```
SELECT DATEDIFF(MONTH, '2023-01-01', GETDATE()) AS
MESES_DESDE_INICIO_ANO;
```

Com a função datepart, posso pegar o dia de hoje, do momento que eu executo a função:

```
SELECT DATEPART(DAY, GETDATE()) AS DIA_DE_HOJE;
```

```
SELECT GETDATE() AS DATA_HOJE, DATEPART(DAY, GETDATE()) AS
DIA_DE_HOJE;
```

Eu posso, por exemplo, testar para saber se uma data é válida ou não:

```
SELECT ISDATE(DATETIMEFROMPARTS(2022, 02, 28, 00, 00, 00, 00));
```

Calculando o número de anos que uma pessoa viveu desde o seu nascimento?

```
SELECT NOME,
DATEDIFF(YEAR, DATA_DE_NASCIMENTO, GETDATE()) AS
NUMERO_DE_ANOS_DE_VIDA
FROM TABELA_DE_CLIENTES;
```

Calculando o número de anos que a pessoa de cpf 1471156710 viveu desde o seu nascimento?

```
SELECT NOME,
DATEDIFF(YEAR, DATA_DE_NASCIMENTO, GETDATE()) AS
NUMERO_DE_ANOS_DE_VIDA
FROM TABELA_DE_CLIENTES
WHERE CPF = '1471156710';
```

DESAFIO DATA POR EXTENSO

Consulta que retornar o nome do cliente e sua data de nascimento por extenso dia, dia da semana, mês e ano

```
SELECT NOME + ' nasceu em ' +
DATENAME (WEEKDAY,DATA_DE_NASCIMENTO) + ',' +
DATENAME (DAY,DATA_DE_NASCIMENTO) + ' de ' +
DATENAME(MONTH, DATA_DE_NASCIMENTO) + ' de ' +
DATENAME(YEAR, DATA_DE_NASCIMENTO) AS DATA_EXTENSO
FROM TABELA_DE_CLIENTES;
```

FUNÇÕES NUMÉRICAS

ROUND - Retorna um valor numérico, arredondado, para o comprimento ou precisão especificados
O ROUND, que retorna um valor numérico arredondado, usando o comprimento de precisão de arredondamento, passado como segundo parâmetro da função.
Então por exemplo, se eu quiser arredondar o número 32,23332 com duas casas decimais:

ROUND(32.23332, 2)

Ele vai me retornar o número 32,23. As outras casas decimais ele vai zerar.

CEILING - retorna o menor inteiro maior que ou igual à expressão numérica identificada. Então, se eu tiver o número 32,23332 e eu aplicar o CEILING:

CEILING(32.23332)

A parte inteira é 32, então o resultado vai ser 33, que é o menor inteiro, maior do que o inteiro existente.
Então se o inteiro existente é 32, o resultado vai ser 33.

Já a função CEILING() desempenha a função oposta, arredondando um número para cima e aproximando-o ao inteiro subsequente. Por exemplo, ao empregar
CEILING(1.6), o retorno será 2.

FLOOR - vai fazer o contrário do CEILING, ele vai retornar o maior inteiro, menor ou igual que a expressão numérica especificada. Então se eu tenho:

FLOOR(32,23332)

O menor inteiro vai ser o próprio 32.

A função FLOOR() realiza um arredondamento para baixo, levando o número ao inteiro imediatamente inferior. Por exemplo, ao aplicar FLOOR(1.6), o resultado será 1.

POWER - vai retornar a potência, vai ser um número elevado a um fator. Então se eu fizer:

POWER(2, 10)

É a mesma coisa que 2 elevado a 10, que vai dar o resultado de 1024.

EXP - Retorna o valor exponencial da expressão float especificada.
O EXP vai retornar o valor exponencial da expressão passada por parâmetro para a função. É baseada na constante e (2,71828182845905), que é a base dos logaritmos naturais.
O expoente de um número é a constante e elevada à potência do número passado por parâmetro.
Por exemplo:

EXP(10)

Eu vou pegar aquele o número 2,71828182845905 e elevá-lo a 10, tendo 22026,4657948067 como resultado. Então, para fórmulas matemáticas,

essa função é super importante.

SQRT - resolve a raiz quadrada do valor flutuante que foi passado por parâmetro para a função. Então, se eu fizer:
SQRT(144)

O resultado será 12, porque a raiz quadrada de 144 é 12.

SIGN - Retorna o sinal positivo (+1), zero (0) ou sinal negativo (-1) da expressão especificada
O SIGN vai retornar um sinal positivo ou negativo, dependendo do sinal do número passado por parâmetro para a função. Então se o número for positivo, ele vai retornar 1, se o número é negativo ele vai retornar -1:

SIGN(-10)

Como -10 é negativo, essa função retornará o número -1.

ABS - vai retornar o valor absoluto(valor positivo do número) e sempre positivo da expressão numérica que foi passada por parâmetro para a função.
ABS altera valores negativos para valores positivos.
ABS não tem efeito em valores zero ou positivos
Então se eu tenho:

ABS(-10)

A função vai retornar o número 10. Então, sempre vai ser o valor absoluto, o valor positivo do número.

PERCENT - é uma função que calcula o resto da divisão entre dois números: dividendo % divisor

Assim, eu tenho resto.
Por exemplo: 10 % 3 é o resto da divisão de 10 por 3, que será 1.

Eu tenho uma série de outras funções que eu posso usar no SQL Server, relacionados com outras partes da matemática ou geometria:

LOG, que retorna o logaritmo natural de uma expressão float, passada por parâmetro para a função.

LOG10, que retorna o logaritmo na base 10 da expressão float passada por parâmetro para a função.

Funções aritméticas como: **ACOS**, **ATAN**, **ASIN**, **COS**, **TAN** e **SIN**.

PI que retorna o número PI, que é muito importante na geometria, principalmente para calcular raios ou áreas de círculo etc.

Estou fazendo o arredondamento dos números

```
SELECT ROUND(3.437,2),ROUND(3.433,2);
```

Consulta que vai retornar o maior inteiro, depois do inteiro do número
O inteiro do número e 3, então ele exibe 4

```
SELECT CEILING(3.433);
```

Consulta que vai retornar o próprio número inteiro do número, ou seja,
nesse caso vai retornar o 3

```
SELECT FLOOR(3.433);
```

```
SELECT POWER(12, 2);
```

```
SELECT EXP(3);
```

Raiz quadrada de 300

```
SELECT SQRT(300);
```

Converter qualquer número negativo em positivo

```
SELECT ABS(-10);
```

DESAFIO: formato do faturamento

Na tabela de notas fiscais, temos o valor do imposto.
Já na tabela de itens, temos a quantidade e o faturamento.

Calcule o valor do imposto pago no ano de 2016, arredondando
para o menor inteiro.

```
SELECT * FROM NOTAS_FISCAIS;
SELECT * FROM ITENS_NOTAS_FISCAIS;

SELECT YEAR(DATA_VENDA) AS ANO, FLOOR(SUM(IMPOSTO * (QUANTIDADE * PRECO)))
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF ON NF.NUMERO = INF.NUMERO
WHERE YEAR(DATA_VENDA) = 2016
GROUP BY YEAR(DATA_VENDA);

SELECT YEAR(NF.DATA_VENDA) AS ANO,
FLOOR(SUM((INF.QUANTIDADE * INF.PRECO) * NF.IMPOSTO)) AS IMPOSTO_PAGO
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
WHERE DATA_VENDA BETWEEN '2016-01-01' AND '2016-12-31'
GROUP BY YEAR(DATA_VENDA);

SELECT YEAR(NF.DATA_VENDA), FLOOR(SUM(NF.IMPOSTO*INF.QUANTIDADE*INF.PRECO)) AS
IMPOSTO_TOTAL
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
```

```
ON NF.NUMERO = INF.NUMERO
WHERE YEAR(NF.DATA_VENDA)=2016
GROUP BY YEAR(NF.DATA_VENDA)
```

Funções de Conversão

CAST e CONVERT - Essas funções convertem uma expressão de um tipo de dados em outro.

Conversão de data para texto

No caso da conversão de data para texto, eu uso a função **CONVERT**.

E por que a conversão de data para texto é importante?

Porque muitas vezes eu quero representar a data de uma maneira diferente do que a data é representada dentro do SQL Server.
Lembra que a data no SQL Server é representada da seguinte forma:
ANO (4 dígitos)-MÊS (2 dígitos)-DIA (2 dígitos).

E nem sempre, quando eu quero executar uma consulta SQL, eu quero exibir a data neste formato.
Eu quero mudar o formato de exibição, então isso é uma conversão de data para texto, usando uma determinada máscara.

E como é que eu defino essa máscara para o SQL Server?

Através de uma numeração.
Existe um número que pode começar do 1 ou do 100 e ele vai crescente, e cada número desse vai representar uma máscara de saída.

Link para consultar

https://learn.microsoft.com/pt-br/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-ver16

CAST - posso usar para fazer conversão de números para números.
Então eu posso pegar um número, converter de inteiro para money, de money para decimal, de decimal para float e assim por diante.

OBSERVAÇÕES:

Para conversões explícitas, a própria instrução determina o tipo de dado resultante, então, quando eu faço a conversão explícita, eu não preciso dizer qual é o tipo de dado que está sendo convertido.
Já para conversões implícitas, eu tenho que atribuir qual é o tipo de dado que eu quero converter.

Estou convertendo a data atual em texto

Como eu estou convertendo uma data para texto, preciso especificar o tipo desse texto.

Por exemplo VARCHAR(10).

E especificamos a máscara 121 (yyyy-mm-dd hh:mi:ss.mmm).

Note que eu só estou vendo o dia, por quê?
Porque o tamanho da máscara é de 23 caracteres "(yyyy-mm-dd hh:mi:ss.mmm)", mas nós definimos um VARCHAR(10), então ele só exibiu os dez primeiros caracteres dessa máscara.

```
SELECT CONVERT(VARCHAR(10), GETDATE(), 121);
```

Para utilizar essa máscara 121, preciso aumentar o tamanho do VARCHAR, como, por exemplo, 25 caracteres:
Quando eu vou converter uma data para texto, é importante que eu coloque a definição do texto com o número de caracteres que coincida com o tamanho final da máscara.

```
SELECT CONVERT(VARCHAR(25), GETDATE(), 121);
```

```
USE SUCOS_FRUTAS;
GO
SELECT * FROM TABELA_DE_CLIENTES;
```

Nós temos o campo DATA_DE_NASCIMENTO, que é a data de nascimento do cliente.
Podemos converter essa data:

Se eu executar isso daqui, teremos um resultado diferente do esperado.
Colocamos o tamanho do VARCHAR como 25, então por
que a hora, o minuto e o segundo não foram exibidos?".

Porque se nós olharmos o tipo original desse campo
"DATA_DE_NASCIMENTO", ele é um campo date e não datetime.

Então, essa data de nascimento que está na tabela, ela não tem hora, minuto e segundo gravada, porque o tipo date não comporta hora, minuto, segundo e milissegundo. Então não adianta usar uma máscara para exibir hora, minuto, segundo e milissegundo, se o campo que está gravado na tabela não tem isso.

```
SELECT DATA_DE_NASCIMENTO, CONVERT(VARCHAR(25),
DATA_DE_NASCIMENTO, 121)
FROM TABELA_DE_CLIENTES;
```

Nós temos o campo DATA_DE_NASCIMENTO, que é a data de nascimento do cliente.
Podemos converter essa data:
Agora, se nós escolhermos uma outra máscara, por exemplo, a 106 (dd mon yyyy):

Eu consigo ver então a data de nascimento do cliente no formato escolhido, olhando o mês em três letras.

```
SELECT DATA_DE_NASCIMENTO, CONVERT(VARCHAR(25),
DATA_DE_NASCIMENTO, 106)
FROM TABELA_DE_CLIENTES;
```

Outra coisa, por exemplo, eu tenho o nome do produto e o preço de lista na tabela de produtos:

```
SELECT NOME_DO_PRODUTO, PRECO_DE_LISTA FROM
TABELA_DE_PRODUTOS;
```

Digamos que eu queira escrever um texto, em vez de exibir apenas o preço do produto, eu vou escrever um texto dizendo "O preço de lista é tal preço", um texto mais bonito para aparecer na tabela.

Então eu poderia usar a função CONCAT, onde eu escreverei:

Ao executar, note que foi feita uma conversão implícita. Ele automaticamente converteu o preço de lista, que é um float, em texto.

```
SELECT NOME_DO_PRODUTO, CONCAT('O preço de lista é: ', PRECO_DE_LISTA)
AS PRECO
FROM TABELA_DE_PRODUTOS;
```

Se eu quisesse converter de forma explícita, bastaria eu ter feito:

Então eu estou convertendo o preço de lista em um VARCHAR(10).
Isso é uma conversão onde eu estou dizendo qual é o tipo que eu quero escrever.
Não preciso usar o cast, porque se olharmos novamente a tabela de conversões, o float é convertido implicitamente para texto.

Na conversão implícita, eu posso fazer essa conversão de forma automática. Essa regra às vezes não funciona direito, não no caso de data, mas pode ser um tipo especial que não funciona, então a primeira coisa que você pode estar testando é usar o concat: se der erro e não funcionar, você usa o cast, pegando o campo e convertendo para o novo tipo que quiser.

```
SELECT NOME_DO_PRODUTO,
CONCAT('O preço de lista é: ', CAST(PRECO_DE_LISTA AS VARCHAR(10)))
AS PRECO
FROM TABELA_DE_PRODUTOS;
```

Desafio: listando expressão natural

Queremos construir um SQL cujo resultado seja para cada cliente:

"O cliente João da Silva comprou R$ 121222,12 no ano de 2016".
Faça isso somente para o ano de 2016.

Dica: Procure na documentação como se usa a função STR para converter um número FLOAT em texto.

Depois de entender a função SRT podemos construir a seguinte consulta:

```
SELECT 'O cliente ' + TC.NOME + ' comprou R$ ' +
TRIM(STR(SUM(INF.QUANTIDADE * INF.PRECO) ,10,2)) + ' no ano de ' + DATENAME(YEAR,
NF.DATA_VENDA)
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF ON NF.NUMERO = INF.NUMERO
INNER JOIN TABELA_DE_CLIENTES TC ON NF.CPF = TC.CPF
WHERE YEAR(NF.DATA_VENDA) = '2016'
GROUP BY TC.NOME, NF.DATA_VENDA;
```

VENDAS VALIDAS

Eu quero fazer um relatório para poder ver, dentro de um determinado mês, quais foram os clientes que ultrapassaram ou não esse volume de compra. Então esse é o meu problema, alguém chegou para mim e disse: "olha, me vê dentro de um mês quais foram os clientes que compraram mais do que o volume de compra especificado para eles".

Inicialmente calculando o volume de venda por cliente.

Então a primeira coisa que eu vou fazer: eu vou selecionar o CPF, que é o cliente, e a data da venda, ambos campos da tabela de notas fiscais, e vou selecionar a quantidade que eu vendi, que estão na tabela de itens das notas fiscais. Logo, como eu preciso selecionar três campos, sendo que um deles está em uma tabela e dois em outra, o que eu preciso fazer um inner join.

O campo em comum entre essas duas tabelas?

É o campo NUMERO, é o que liga essas duas tabelas, então ele vai ser o critério do join:
Na tabela de notas fiscais, eu quero ver o CPF e a data da venda e da tabela de itens, eu quero ver a quantidade

Tenho quantas vendas foi realizada na data e por qual pessoa

```
SELECT NF.CPF, NF.DATA_VENDA, INF.QUANTIDADE
FROM NOTAS_FISCAIS AS NF
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON NF.NUMERO = INF.NUMERO;
```

Eu preciso ter essa informação dentro do mês e do ano.
Então a primeira coisa que eu vou fazer é escrever a data só com mês e ano. Nós podemos usar o "convert" para isso, utilizando a máscara 120 (yyyy-mm-dd hh:mi:ss) e limitando o tamanho do VARCHAR para 7, assim que só exibirei o ano e o mês:

Nesse caso quero que mostre somente o mês e o ano da venda

```
SELECT NF.CPF,
CONVERT (VARCHAR(7),NF.DATA_VENDA, 120) AS MES_ANO,
INF.QUANTIDADE
FROM NOTAS_FISCAIS AS NF
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON NF.NUMERO = INF.NUMERO;
```

Mas eu preciso ter isso agrupado, porque eu preciso calcular a soma total das quantidades dentro do mês e do ano, porque eu preciso comparar com o volume do cadastro, que é o volume que está, por contrato, acertado para o cliente comprar no mês.

Nós vamos fazer aqui um SUM, eu vou somar a quantidade, chamando de quantidade total:

A partir do momento que eu coloquei um SUM, eu sou obrigado a usar um group by, mas por quais campos?

Pela chave, que é onde eu vou somar, que é o CPF e pela data da venda, convertida para mês e ano.

```
SELECT NF.CPF,
CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO,
SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
FROM NOTAS_FISCAIS AS NF
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON NF.NUMERO = INF.NUMERO
GROUP BY
NF.CPF, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120);
```

O cpf 9283760794 comprou no mês_ ano de 2016-04 a quantidade total de 23352 litros de suco

Vamos olhar a tabela de clientes, onde eu tenho o CPF, o nome do cliente e o volume de compra:

```
SELECT CPF, NOME, VOLUME_DE_COMPRA
FROM TABELA_DE_CLIENTES;
```

Preciso comparar o volume de compra que foi acertado com o cliente no início do contrato.
Nesse caso estou verificando com a consulta abaixo essa situação

```
SELECT CPF, NOME, VOLUME_DE_COMPRA FROM
TABELA_DE_CLIENTES;
```

Com essa consulta abaixo estou verificando a quantidade de vendas que o cliente fez nos meses e anos.

```
SELECT NF.CPF,
CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO,
SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
FROM NOTAS_FISCAIS AS NF
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON NF.NUMERO = INF.NUMERO
GROUP BY
NF.CPF, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120);
```

Isso tudo para verificar se o cliente não comprou mais que o combinado no início do contrato

Para fazer uma consulta que retorne o volume de compra que foi combinado no contrato, com a quantidade de vendas que o cliente está comprando.
Utilizamos as subqueries e fazemos um join de subqueries.
O campo em comum das duas consultas e o CPF do cliente.

O inner join não será com uma tabela, mas com a query que fizemos anteriormente:
Eu preciso dar um alias a esta subquery, então eu vou chamar de TV, de total de vendas:

No join, o campo que vai fazer a ligação será o CPF, mas não podemos chamar por NF.CPF, porque como o on está sendo visualizado fora da subquery, ele tem como alias TV
Vamos selecionar primeiro os campos da tabela de clientes: CPF, NOME, VOLUME_DE_COMPRA.
E da tabela que é subquery, que tem o alias TV, eu vou selecionar os campos MES_ANO e QUANTIDADE_TOTAL

```
SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL
FROM TABELA_DE_CLIENTES AS TC
INNER JOIN (
    SELECT NF.CPF,
    CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO,
    SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
    FROM NOTAS_FISCAIS NF
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON NF.NUMERO = INF.NUMERO
    GROUP BY NF.CPF, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) AS TV
ON TV.CPF = TC.CPF;
```

Depois de executar a consulta percebemos que no mês 04 do ano 2016 no contrato (VOLUME_DE_COMPRA) era 25000, o cliente comprou nesse mês 23352 litros de suco, ou sejá, menos do que foi combinado no contrato

Alguns compraram menos outros mais do que foi estipulado no contrato

Vamos fazer uma classificação para especificar quais são os clientes

que estão dentro ou que estão fora do limite especificado no contrato.

Se o volume total (QUANTIDADE_TOTAL) for maior que o volume de compra(VOLUME_DE_COMPRA) ,eu vou dizer, por exemplo, que as vendas foram inválidas.
Se o volume de compra(VOLUME_DE_COMPRA) for maior ou igual que o volume total (QUANTIDADE_TOTAL) as vendas foram válidas:

```
 SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL,
(CASE WHEN TC.VOLUME_DE_COMPRA >= TV.QUANTIDADE_TOTAL THEN 'VENDAS VÁLIDAS'
ELSE 'VENDAS INVÁLIDAS' END) AS RESULTADO
FROM TABELA_DE_CLIENTES AS TC
INNER JOIN (
   SELECT NF.CPF,
   CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO,
   SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
   FROM NOTAS_FISCAIS AS NF
   INNER JOIN ITENS_NOTAS_FISCAIS AS INF
   ON NF.NUMERO = INF.NUMERO
   GROUP BY NF.CPF, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) AS TV
ON TV.CPF = TC.CPF;
```

Vamos filtrar essa consulta por mês e ano, porque eu não vou fazer isso para o ano todo.

Se rodarmos essa consulta, vamos ver que, dentro do mês de Janeiro de 2015, os clientes que compraram produtos e eu consigo ver quais foram válidos e quais foram inválidos. Agora, eu vou apresentar esse relatório para o meu cliente e aí ele vai conseguir determinar quais foram os clientes que compraram dentro do volume de compras estipulado pelo contrato e os que compraram fora daquele volume de compras estipulado pelo contrato.

```
SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL,
(CASE WHEN TC.VOLUME_DE_COMPRA >= TV.QUANTIDADE_TOTAL THEN 'VENDAS
VÁLIDAS'
ELSE 'VENDAS INVÁLIDAS' END) AS RESULTADO
FROM TABELA_DE_CLIENTES AS TC
INNER JOIN (
   SELECT NF.CPF,
   CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO,
   SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
   FROM NOTAS_FISCAIS AS NF
   INNER JOIN ITENS_NOTAS_FISCAIS AS INF
   ON NF.NUMERO = INF.NUMERO
   GROUP BY NF.CPF, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) AS TV
ON TV.CPF = TC.CPF
WHERE TV.MES_ANO = '2015-01';
```

Desafio: complementando o relatório

Construímos um relatório que apresentou os clientes que tiveram vendas válidas e inválidas.

Nesse sentido, nosso compromisso agora é que você complemente este relatório, isto é, listando somente os que tiveram vendas inválidas e calculando a diferença entre o limite de venda máximo e o realizado, em percentuais.

Vamos fazer um filtro especial para somente os que tiveram seus limites estourados, ou sejá, vai mostrar somente os que tiveram vendas inválidas, que ultrapassaram o valor combinado no contrato.

```
SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL,
(CASE WHEN TC.VOLUME_DE_COMPRA >= TV.QUANTIDADE_TOTAL THEN 'VENDAS VÁLIDAS'
ELSE 'VENDAS INVÁLIDAS' END) AS RESULTADO
FROM TABELA_DE_CLIENTES TC
INNER JOIN
(
SELECT
NF.CPF
,CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO
,SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
GROUP BY
NF.CPF
, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) TV
ON TV.CPF = TC.CPF
WHERE TV.MES_ANO = '2015-01'
AND (TC.VOLUME_DE_COMPRA - TV.QUANTIDADE_TOTAL) < 0
```

E finalmente, então, inserimos o novo indicador.

```
SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL,
(CASE WHEN TC.VOLUME_DE_COMPRA >= TV.QUANTIDADE_TOTAL THEN 'VENDAS VÁLIDAS'
ELSE 'VENDAS INVÁLIDAS' END) AS RESULTADO,
(1 - (TC.VOLUME_DE_COMPRA/TV.QUANTIDADE_TOTAL)) * 100 AS PERCENTUAL
FROM TABELA_DE_CLIENTES TC
INNER JOIN
(
SELECT
NF.CPF
,CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO
,SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
GROUP BY
NF.CPF
, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) TV
ON TV.CPF = TC.CPF
WHERE TV.MES_ANO = '2015-01'
AND (TC.VOLUME_DE_COMPRA - TV.QUANTIDADE_TOTAL) < 0;
```

Arredondar o resultado para duas casas decimais.

```
SELECT
TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA, TV.MES_ANO, TV.QUANTIDADE_TOTAL,
```

```
(CASE WHEN TC.VOLUME_DE_COMPRA >= TV.QUANTIDADE_TOTAL THEN 'VENDAS VÁLIDAS'
ELSE 'VENDAS INVÁLIDAS' END) AS RESULTADO,
ROUND((1 - (TC.VOLUME_DE_COMPRA/TV.QUANTIDADE_TOTAL)) * 100, 2) AS PERCENTUAL
FROM TABELA_DE_CLIENTES TC
INNER JOIN
(
SELECT
NF.CPF
,CONVERT(VARCHAR(7), NF.DATA_VENDA, 120) AS MES_ANO
,SUM(INF.QUANTIDADE) AS QUANTIDADE_TOTAL
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
GROUP BY
NF.CPF
, CONVERT(VARCHAR(7), NF.DATA_VENDA, 120)
) TV
ON TV.CPF = TC.CPF
WHERE TV.MES_ANO = '2015-01'
AND (TC.VOLUME_DE_COMPRA - TV.QUANTIDADE_TOTAL) < 0;
```

Vendas por sabor

O usuário pediu um outro relatório, ele quer ver as vendas anuais, ou seja, as vendas dentro do ano, dos meus sucos de frutas por sabor.
Mas não é só isso que ele quer não, ele quer que eu apresente esse relatório ordenado, do que mais vendeu para o que menos vendeu.

Para eu ter as vendas por sabor, eu preciso ter o sabor, que está na tabela de produtos, preciso ter a quantidade, que está na tabela dos itens das notas fiscais e preciso ter a data da venda, que está na tabela de notas fiscais.
Afinal, eu quero ver essa venda fechada no ano.

Eu tenho um campo em cada tabela, então têm três tabelas que eu tenho que fazer o meu join.
Nós podemos fazer join entre mais do que duas tabelas.
Então precisamos juntar essas três tabelas e pegar esses três campos,

Eu vou criar uma nova consulta e nós vamos fazer um join entre três tabelas

Primeira coisa: a tabela de produtos tem uma ligação com a tabela de itens de notas fiscais, através do campo CODIGO_DO_PRODUTO, que é o campo em comum entre essas duas tabelas:

Mas agora eu preciso ligar a tabela de itens com a tabela de notas. O campo em comum é o NUMERO:

Eu preciso buscar o campo SABOR da tabela de produtos, o campo QUANTIDADE da tabela de itens e o campo DATA_VENDA de notas fiscais:

```
SELECT TP.SABOR, NF.DATA_VENDA, INF.QUANTIDADE
FROM TABELA_DE_PRODUTOS AS TP
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS AS NF
ON INF.NUMERO = NF.NUMERO;
```

o meu usuário quer ver isso por ano. Então eu preciso agrupar:
Eu tenho a minha venda agora por ano,

```
SELECT TP.SABOR,
YEAR(NF.DATA_VENDA) AS ANO,
SUM(INF.QUANTIDADE) AS VENDA_ANO
FROM TABELA_DE_PRODUTOS AS TP
INNER JOIN ITENS_NOTAS_FISCAIS AS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS AS NF
ON INF.NUMERO = NF.NUMERO
GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA);
```

MAS preciso filtrar isso para um ano específico, porque ele não quer ver todos os anos, ele quer ver isso ano a ano.

Então vamos supor que eu vá selecionar o ano de 2015, eu preciso colocar um where, lembrando que o where não fica depois do group by, ele fica antes

```
SELECT TP.SABOR,
YEAR(NF.DATA_VENDA) AS ANO,
SUM(INF.QUANTIDADE) AS VENDA_ANO
FROM TABELA_DE_PRODUTOS TP
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS NF
ON INF.NUMERO = NF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2015
GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA);
```

Só que o usuário quer isso ordenado da maior venda para a menor venda.
Então para nós chegarmos a essa posição, colocamos um order by:

```
SELECT TP.SABOR,
YEAR(NF.DATA_VENDA) AS ANO,
SUM(INF.QUANTIDADE) AS VENDA_ANO
FROM TABELA_DE_PRODUTOS TP
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS NF
ON INF.NUMERO = NF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2015
GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
```

```
ORDER BY SUM(INF.QUANTIDADE) DESC;
```

Então eu tenho um ranking, no ano de 2015, manga foi o sabor que mais vendeu, vendeu 601 mil, depois vem laranja, melancia e assim por diante.

Eu entreguei isso para o meu cliente, só que ele falou: "gostei, mas eu queria uma coisa mais rebuscada, eu quero colocar ter um percentual da participação desta venda em relação à venda total do ano".

Então por exemplo, se eu vender 1 milhão de litros e se laranja vender 100 mil, laranja representou 10% das vendas do ano.

Então o relatório voltou para mim, porque tenho que colocar agora um percentual, como é que eu vou colocar e calcular esse percentual?

Vamos fazer o seguinte: primeiro vamos nos preocupar em calcular esse total, eu vou agora fazer uma segunda query, onde eu vou calcular o total do ano:

Para calcular o total do ano, preciso ter a quantidade, que estão na tabela de itens, e preciso ter a data da venda, que está na tabela de notas fiscais, e aí juntar essas duas tabelas através do campo NUMERO:

Como eu fiz um SUM, eu preciso de um group by

E eu vou colocar o where, para filtrar o ano, para mostrar somente 2015:
Se eu rodar essa segunda seleção, eu tenho aqui o total do ano.

```
SELECT YEAR(NF.DATA_VENDA) AS ANO,
SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2015
GROUP BY YEAR(NF.DATA_VENDA);
```

Eu tenho duas consultas separadas, uma que me mostra o ranking do valor e outra que mostra o total.

Para eu calcular a participação, eu preciso pegar o número de vendas por sabor no ano, dividir pelo número de vendas totais do ano e multiplicar por 100, aí eu vou ter a participação do sabor em relação ao total.

Como é que eu consigo colocar juntar esses dois números?

Com o join

E por qual campo em comum?

O campo ano.

Então vamos lá, vamos fazer agora aqui um outro join, onde meu primeiro from será a primeira query que fizemos:
E o inner join será a segunda query, que acabamos de fazer:
Eu vou chamar a primeira query de VS, que tem a ver com venda por sabor e a segunda query eu vou chamar de VA, que é a venda no ano. E o campo em comum é o campo ano, da consulta VS, com o campo ano, da consulta VA:
Da tabela VS, eu vou selecionar os campos SABOR, ANO e VENDA_ANO,
e da tabela VA, eu vou selecionar o campo VENDA_TOTAL_ANO

Note o erro, pois eu estou usando um order by dentro de uma subquery,
isso não pode ser feito.

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO, VA.VENDA_TOTAL_ANO
FROM (
   SELECT TP.SABOR,
   YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_ANO
   FROM TABELA_DE_PRODUTOS TP
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
   INNER JOIN NOTAS_FISCAIS NF
   ON INF.NUMERO = NF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
   ORDER BY SUM(INF.QUANTIDADE) DESC
) VS
INNER JOIN (
   SELECT YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
   FROM NOTAS_FISCAIS NF
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON NF.NUMERO = INF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO;
```

vou tirar o order by da subquery e ordenar a query final,
porém não vamos mais ordenar pelo campo que estávamos ordenando
antes, e sim ordenar pelo campo VS.VENDA_ANO:

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO, VA.VENDA_TOTAL_ANO
FROM (
   SELECT TP.SABOR,
   YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_ANO
   FROM TABELA_DE_PRODUTOS TP
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
   INNER JOIN NOTAS_FISCAIS NF
   ON INF.NUMERO = NF.NUMERO
```

```
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN (
    SELECT YEAR(NF.DATA_VENDA) AS ANO,
    SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
    FROM NOTAS_FISCAIS NF
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON NF.NUMERO = INF.NUMERO
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Agora, para calcular o percentual, eu tenho que dividir VENDA_ANO por VENDA_TOTAL_ANO e multiplicar por 100:

Ao executar, vemos a coluna inteira com o valor 0, por que isso está acontecendo?

É um problema de tipo de campo, pois eu estou dividindo VS.VENDA_ANO por VA.VENDA_TOTAL_ANO, e esses dois valores estão vindo do campo INF.QUANTIDADE.

O que acontece é que o campo QUANTIDADE é um inteiro. Então o que acontece?

Ele deveria ter sido criado como float,quando eu pego um inteiro e divido por outro inteiro, o SQL, se der um número menor do que 1, que é o nosso caso, porque eu estou dividindo a venda do sabor pela venda total, então sempre o numerador vai ser menor que o denominador nessa divisão, logo, o valor vai estar sempre o que? Menor que um.

Como não tem número inteiro, o SQL obrigatoriamente me dá como resultado um outro inteiro. E aí como esse valor dá menor que 1, eu só vejo 0.

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO, VA.VENDA_TOTAL_ANO,
   (VS.VENDA_ANO/VA.VENDA_TOTAL_ANO) * 100 AS PERCENTUAL
FROM (
   SELECT TP.SABOR,
   YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_ANO
   FROM TABELA_DE_PRODUTOS TP
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
   INNER JOIN NOTAS_FISCAIS NF
   ON INF.NUMERO = NF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN (
   SELECT YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
   FROM NOTAS_FISCAIS NF
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON NF.NUMERO = INF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Para resolver esse problema, eu tenho que converter este campo para float:

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO, VA.VENDA_TOTAL_ANO,
   (CONVERT(FLOAT, VS.VENDA_ANO) / CONVERT(FLOAT,
VA.VENDA_TOTAL_ANO)) * 100 AS PERCENTUAL
FROM (
   SELECT TP.SABOR,
   YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_ANO
   FROM TABELA_DE_PRODUTOS TP
   INNER JOIN ITENS_NOTAS_FISCAIS INF
   ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
   INNER JOIN NOTAS_FISCAIS NF
   ON INF.NUMERO = NF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN (
   SELECT YEAR(NF.DATA_VENDA) AS ANO,
   SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
   FROM NOTAS_FISCAIS NF
   INNER JOIN ITENS_NOTAS_FISCAIS INF
```

```
   ON NF.NUMERO = INF.NUMERO
   WHERE YEAR(NF.DATA_VENDA) = 2015
   GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Se eu rodar agora a minha consulta, vemos que resolveu.
Só que o meu usuário quer ver o percentual de participação só com duas casas decimais. Então, sobre o valor todo, eu vou aplicar o meu round com duas casas decimais

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO, VA.VENDA_TOTAL_ANO,
    ROUND((CONVERT(FLOAT, VS.VENDA_ANO) / CONVERT(FLOAT, VA.VENDA_TOTAL_ANO)) *
100, 2) AS PERCENTUAL
FROM (
    SELECT TP.SABOR,
    YEAR(NF.DATA_VENDA) AS ANO,
    SUM(INF.QUANTIDADE) AS VENDA_ANO
    FROM TABELA_DE_PRODUTOS TP
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
    INNER JOIN NOTAS_FISCAIS NF
    ON INF.NUMERO = NF.NUMERO
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN (
    SELECT YEAR(NF.DATA_VENDA) AS ANO,
    SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
    FROM NOTAS_FISCAIS NF
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON NF.NUMERO = INF.NUMERO
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

O indicador VA.VENDA_TOTAL_ANO não precisa aparecer mais, pois eu não preciso ter o valor da venda total do ano repetido na coluna, então esse aqui eu tirar, ficando o resultado final da seguinte forma

```
SELECT VS.SABOR, VS.ANO, VS.VENDA_ANO,
    ROUND((CONVERT(FLOAT, VS.VENDA_ANO) / CONVERT(FLOAT, VA.VENDA_TOTAL_ANO)) *
100, 2) AS PERCENTUAL
FROM (
    SELECT TP.SABOR,
    YEAR(NF.DATA_VENDA) AS ANO,
    SUM(INF.QUANTIDADE) AS VENDA_ANO
    FROM TABELA_DE_PRODUTOS TP
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
    INNER JOIN NOTAS_FISCAIS NF
    ON INF.NUMERO = NF.NUMERO
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN (
```

```
    SELECT YEAR(NF.DATA_VENDA) AS ANO,
    SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
    FROM NOTAS_FISCAIS NF
    INNER JOIN ITENS_NOTAS_FISCAIS INF
    ON NF.NUMERO = INF.NUMERO
    WHERE YEAR(NF.DATA_VENDA) = 2015
    GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Então eu tenho para 2015, manga representou 16,78% das vendas totais, laranja 13,34%, melancia 13,23% e assim por diante.

Se eu quiser ver isso para o ano de 2016, basta eu colocar 2016 nos dois where e aí executar a consulta de novo.

Desafio: vendas percentuais por tamanho

Agora, a ideia é focar neste relatório novamente, porém que você modifique o relatório tendo como objetivo ver o ranking das vendas por tamanho.

```
SELECT
VS.SABOR, VS.ANO, VS.VENDA_ANO,
ROUND((CONVERT( FLOAT, VS.VENDA_ANO) / CONVERT( FLOAT, VA.VENDA_TOTAL_ANO)) *
100, 2) AS PERCENTUAL
FROM
(
SELECT
TP.SABOR
,YEAR(NF.DATA_VENDA) AS ANO
,SUM(INF.QUANTIDADE) AS VENDA_ANO
FROM TABELA_DE_PRODUTOS TP
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS NF
ON INF.NUMERO = NF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2016
GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN
(
SELECT
YEAR(NF.DATA_VENDA) AS ANO
, SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2016
GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Logo, faça um FIND/REPLACE na consulta substituindo onde temos SABOR por TAMANHO. Ficando assim.

```
SELECT
VS.TAMANHO, VS.ANO, VS.VENDA_ANO,
```

```
ROUND((CONVERT( FLOAT, VS.VENDA_ANO) / CONVERT( FLOAT, VA.VENDA_TOTAL_ANO)) *
100, 2) AS PERCENTUAL
FROM
(
SELECT
TP.TAMANHO
,YEAR(NF.DATA_VENDA) AS ANO
,SUM(INF.QUANTIDADE) AS VENDA_ANO
FROM TABELA_DE_PRODUTOS TP
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO
INNER JOIN NOTAS_FISCAIS NF
ON INF.NUMERO = NF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2016
GROUP BY TP.TAMANHO, YEAR(NF.DATA_VENDA)
) VS
INNER JOIN
(
SELECT
YEAR(NF.DATA_VENDA) AS ANO
, SUM(INF.QUANTIDADE) AS VENDA_TOTAL_ANO
FROM NOTAS_FISCAIS NF
INNER JOIN ITENS_NOTAS_FISCAIS INF
ON NF.NUMERO = INF.NUMERO
WHERE YEAR(NF.DATA_VENDA) = 2016
GROUP BY YEAR(NF.DATA_VENDA)
) VA
ON VS.ANO = VA.ANO
ORDER BY VS.VENDA_ANO DESC;
```

Pronto, agora podemos dar por finalizada nossa demanda de modificar o relatório por ranking das vendas por tamanho.