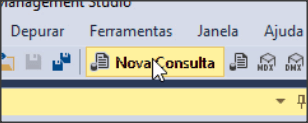
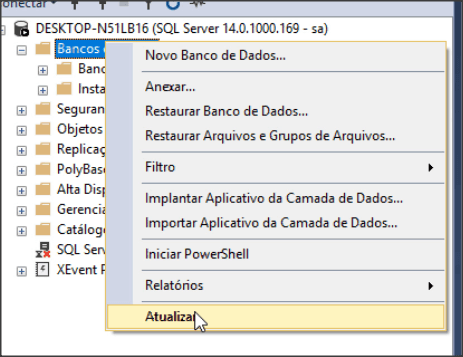
1) No **SQL Server Management Studio 2017**, conecte-se ao seu servidor, e clique sobre **Nova Consulta**:



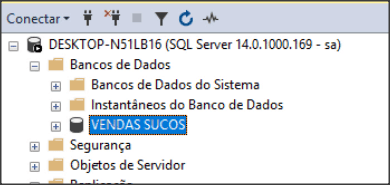
2) Digite o comando abaixo e execute-o:

CREATE DATABASE [VENDAS SUCOS]

3) Volte à pasta **Banco de Dados** e clique com o botão da direita do mouse sobre ela, escolhendo a opção **Atualiza**:



4) Você pode ver o banco criado:



5) Execute comandos mais complexos para criação da base de dados. Digite o exemplo abaixo, mas antes certifique-se que você tenha criado em sua máquina o subdiretório **C:\TEMP\DATA**:

CREATE DATABASE [VENDAS SUCOS 2]

ON (NAME = VENDAS\_SUCOS\_2\_DAT,

FILENAME = 'C:\TEMP\DATA\VENDAS\_SUCOS\_02.MDF',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5)

LOG ON (NAME = VENDAS\_SUCOS\_2\_LOG,

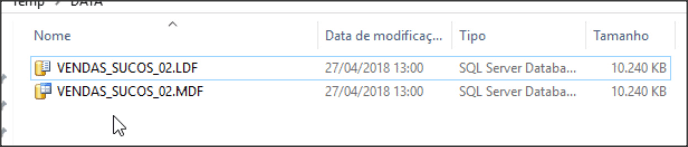
FILENAME = 'C:\TEMP\DATA\VENDAS\_SUCOS\_02.LDF',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5)

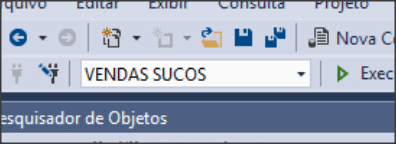
6) Verifique os arquivos criados em **C:\TEMP\DATA**:



7) Para apagar o banco de dados, basta executar o comando **DROP**, como mostrado abaixo:

DROP DATABASE [VENDAS SUCOS 2]

8) Certifique-se que você esteja acessando a base **VENDAS SUCOS**:



9) Para criar a tabela de produtos, conforme a especificação do diagrama de entidades e relacionamentos digite:

CREATE TABLE [PRODUTOS]

([CÓDIGO] VARCHAR(10) NOT NULL,

[DESCRITOR] VARCHAR(100) NULL,

[SABOR] VARCHAR(50) NULL,

[TAMANHO] VARCHAR(50) NULL,

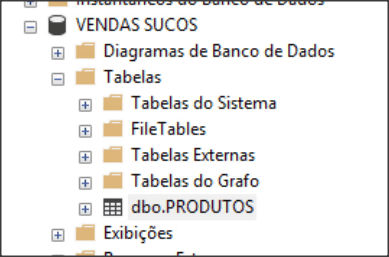
[EMBALAGEM] VARCHAR(50) NULL,

[PREÇO LISTA] FLOAT NULL,

CONSTRAINT [PK\_PRODUTOS]

PRIMARY KEY CLUSTERED ([CÓDIGO]))

10) Execute o comando e atualize a lista de dados da árvore à esquerda no Management Studio. Veja se a tabela foi criada:



11) Repita o processo, criando a tabela de vendedores, digitando o comando abaixo:

CREATE TABLE [VENDEDORES]

([MATRÍCULA] VARCHAR(5) NOT NULL,

[NOME] VARCHAR(100) NULL,

[BAIRRO] VARCHAR(50) NULL,

[COMISSÃO] FLOAT NULL,

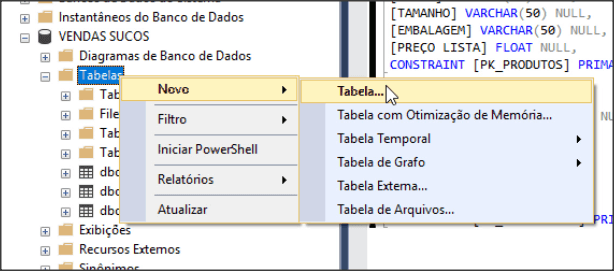
[DATA ADMISSÃO] DATE NULL,

[FÉRIAS] BIT NULL,

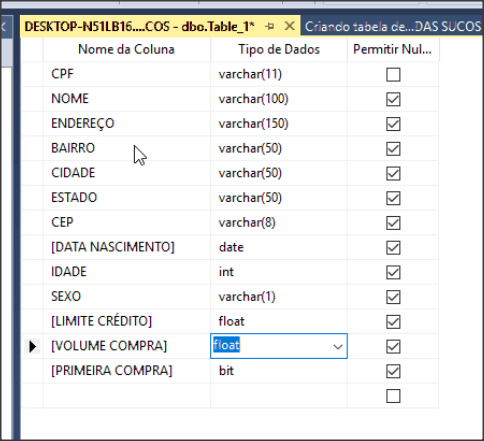
CONSTRAINT [PK\_VENDEDORES]

PRIMARY KEY CLUSTERED ([MATRÍCULA]))

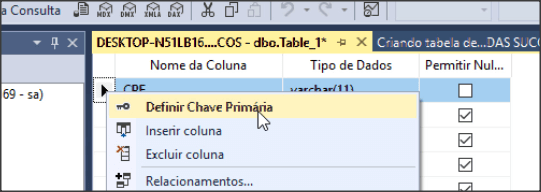
12) Para criar a tabela de clientes pelo assistente, clique com o botão da direita do mouse sobre **Tabelas** e escolha as opções **Novo --> Tabela**:



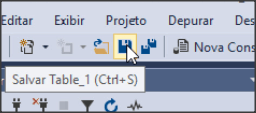
13) No *grid* apresentado, digite os campos da nova tabela de clientes:

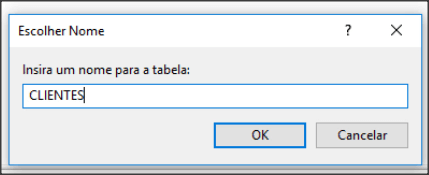


14) Clique com o botão da direita do mouse sobre o primeiro campo **CPF** e selecione a opção do menu **Definir Chave Primária**:

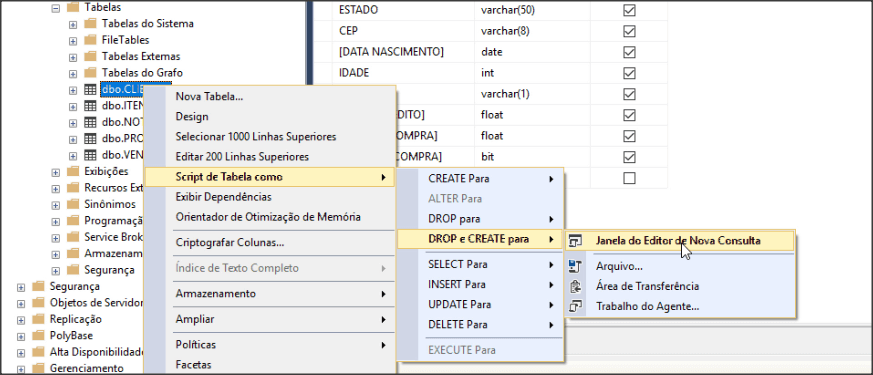


15) Clique no botão **Salvar** e digite o nome da tabela (**CLIENTES**):

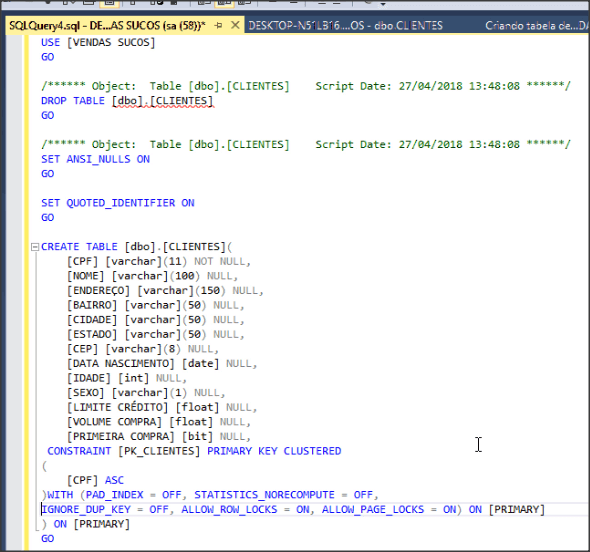




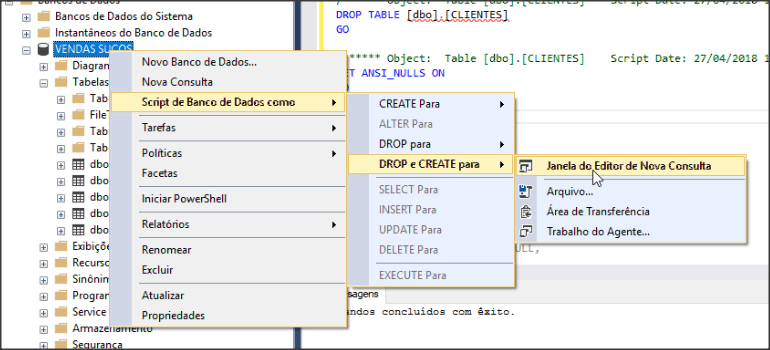
16) Clique com o botão da direita do mouse sobre o nome da tabela **CLIENTES** e escolha a opção **Script de Tabela como --> DROP e CREATE para --> Janela do Editor de Nova Consulta**:



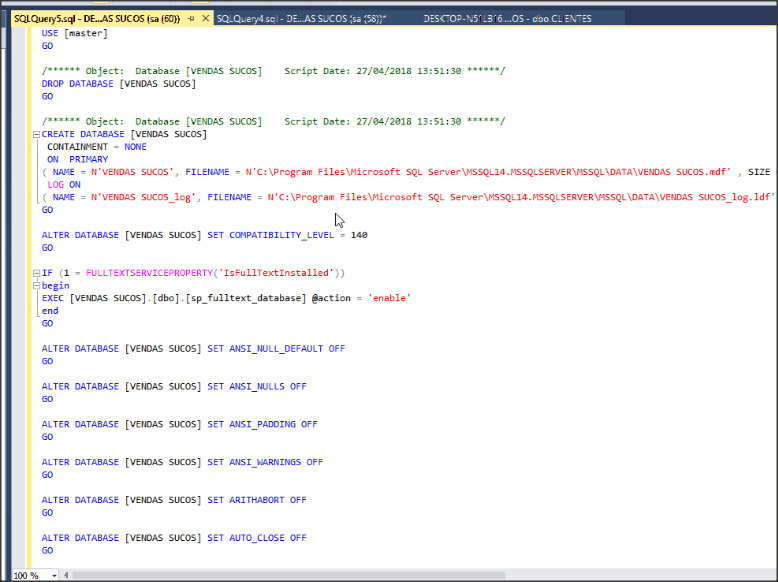
17) Haverá uma janela com o script gerado pelo SQL Server, para criação de uma tabela com as propriedades da tabela **CLIENTES**:



18) Você pode gerar o script de criação do banco de dados clicando com o botão da direita do mouse sobre a base de dados **VENDA SUCOS** e selecionando **Script de Banco de Dados como --> DROP e CREATE para --> Janela do Editor de Nova Consulta**:



19) Você terá:



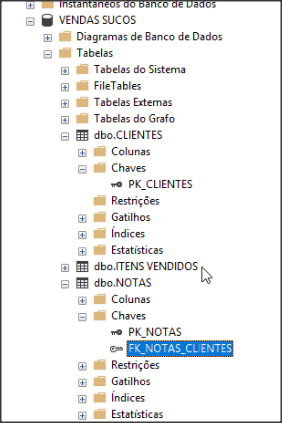
Não execute estes comandos para não perder o seu banco de dados.

20) Através do campo **CPF**, crie um relacionamento entre a tabela de clientes e a de itens de notas fiscais. Para isso, abra uma nova consulta no **SQL Server Management Studio**, certificando-se que o banco **VENDAS SUCOS** esteja selecionado, e digite o comando abaixo:

ALTER TABLE [NOTAS] WITH CHECK ADD CONSTRAINT [FK\_NOTAS\_CLIENTES]

FOREIGN KEY([CPF]) REFERENCES [CLIENTES] ([CPF])

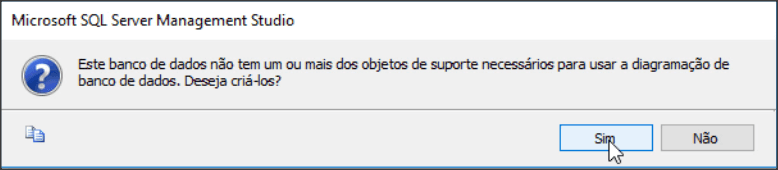
21) Execute este comando. Depois verifique as restrições criadas nas tabelas **CLIENTES** e **NOTAS**, no item da árvore de objetos chamado **Chaves**:



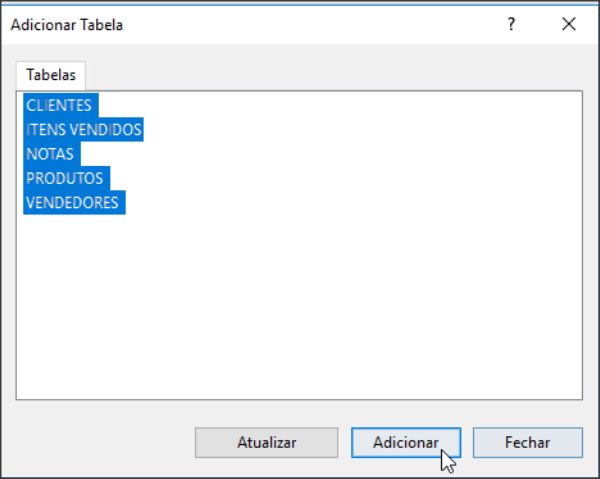
22) Crie um novo relacionamento, agora usando o diagrama do SQL Server 2017. Para isso, clique com o botão da direita do mouse sobre a pasta **Diagramas de Banco de Dados**, abaixo do banco de dados **VENDAS SUCOS**, e selecione a opção **Novo Diagrama de Banco de Dados**:



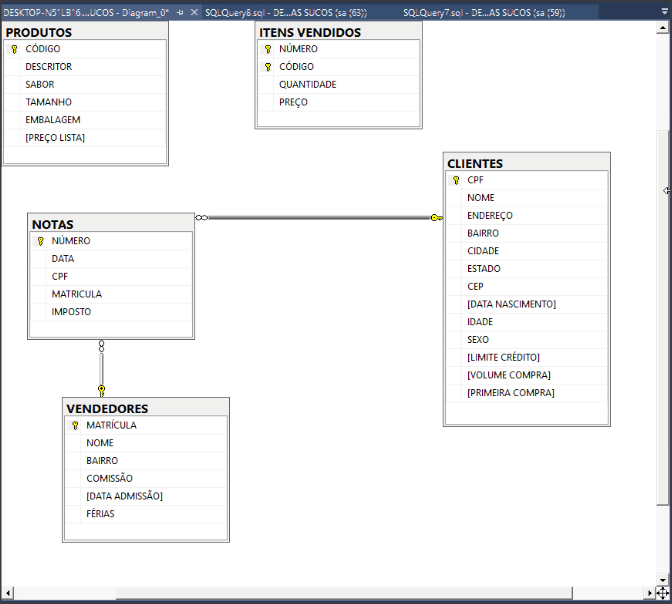
23) Clique em **Sim** na caixa de diálogo abaixo:



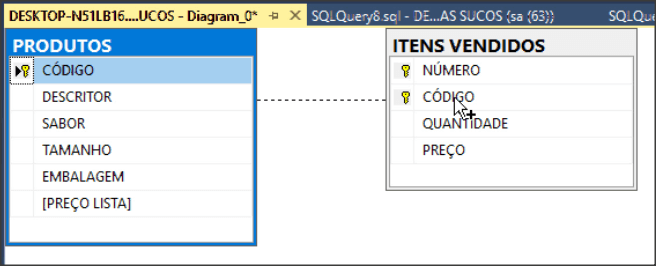
24) Selecione todas as tabelas e depois clique no botão **Adicionar**:



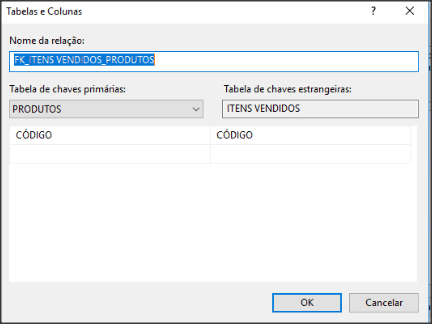
25) Organize o esquema visual das tabelas a fim de poder observar em um único diagrama:



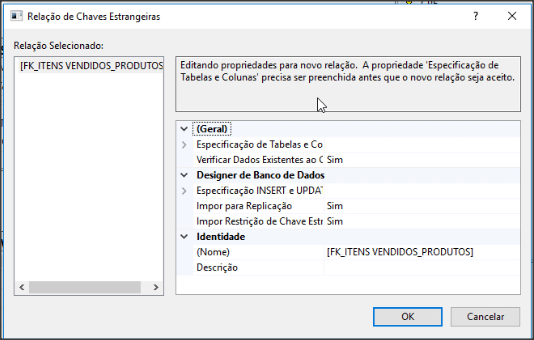
26) Com o mouse, clique no campo **Código**, da tabela **PRODUTOS** e, mantendo o dedo no clique no mouse, sem soltá-lo, arraste-o para o campo **Código**, da tabela **ITENS VENDIDOS**:



27) Após a associação, ao soltar o dedo do botão do mouse, uma caixa de diálogo será mostrada, onde você pode ver a associação que será construída. Depois clique em **OK**:



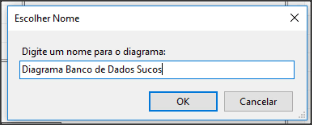
28) Na próxima tela, você pode ver as propriedades do relacionamento. Clique novamente em **OK**:



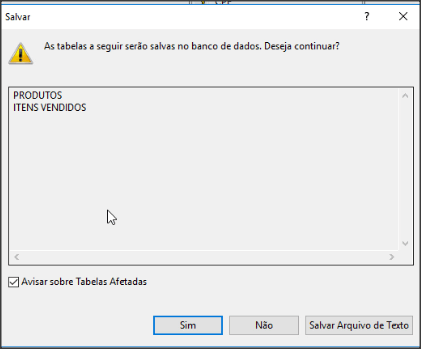
29) Para salvar a criação do relacionamento, clique no botão **Salvar**:



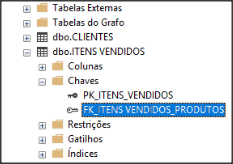
30) O **SQL Server Management Studio** irá perguntar o nome do diagrama, coloque um nome de sua preferência e clique em **OK**:



31) Confirme a alteração que será executada na tabela:



32) Por fim, veja o relacionamento criado:



Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1. Vá no SQL Server Management Studio e crie uma nova consulta associada à base de dados **VENDAS SUCOS**. Digite o comando abaixo:

INSERT INTO [PRODUTOS]

([CÓDIGO],[DESCRITOR],[SABOR],

[TAMANHO],[EMBALAGEM],[PREÇO LISTA])

VALUES ('1040107','Light - 350 ml - Melancia',

'Melancia', '350 ml','Lata', 4.56)

2) Execute mais de um comando de inclusão:

INSERT INTO [PRODUTOS]

([CÓDIGO],[DESCRITOR],[SABOR],

[TAMANHO],[EMBALAGEM],[PREÇO LISTA])

VALUES ('1040108','Light - 350 ml - Graviola',

'Graviola', '350 ml','Lata', 4.00)

INSERT INTO [PRODUTOS]

([CÓDIGO],[DESCRITOR],[SABOR],[TAMANHO],

[EMBALAGEM],[PREÇO LISTA])

VALUES ('1040109','Light - 350 ml - Açai',

'Açai', '350 ml','Lata', 5.60)

Depois de selecionar estes dois comandos e executá-los, as duas inclusões serão efetuadas.

3) Você também pode incluir mais de um registro em um mesmo comando, como mostrado abaixo:

INSERT INTO [PRODUTOS]

([CÓDIGO],[DESCRITOR],[SABOR],[TAMANHO],

[EMBALAGEM],[PREÇO LISTA])

VALUES

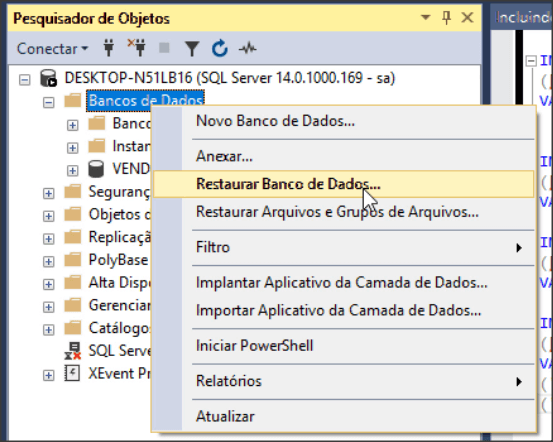
('1040110','Light - 350 ml - Jaca', 'Jaca',

'350 ml','Lata', 6.00),

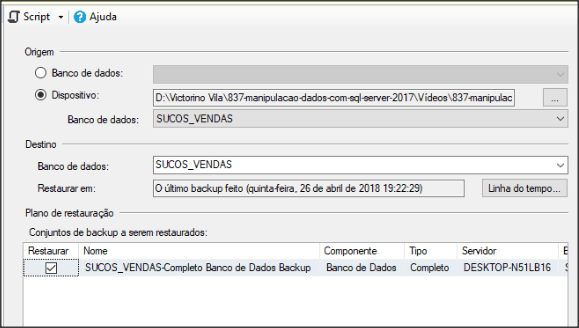
('1040111','Light - 350 ml - Manga', 'Manga',

'350 ml','Lata', 3.50)

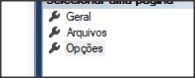
4) Você pode mesclar o comando **INSERT** com o comando **SELECT** para incluir diversos registros de uma só vez. Para isso, carregue o backup de um outro banco de dados, que será usado como fonte. Clique com o botão da direita do mouse sobre a pasta **Banco de Dados** e selecione a opção **Restaurar Banco de Dados**:

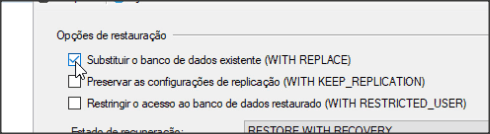


5) Clique na opção **Dispositivo** e selecione o arquivo **SUCOS\_VENDAS.BAK**, que você pode baixar [aqui](https://s3.amazonaws.com/caelum-online-public/837-manipulacao-dados-com-sql-server-2017/03/SUCOS_VENDAS.BAK):

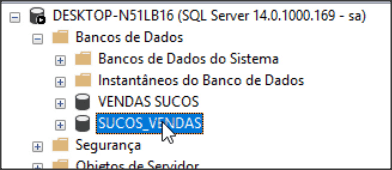


6) Na seção **Opções**, selecione **Substituir o banco de dados existente (WITH REPLACE)**. Depois clique em **OK**:

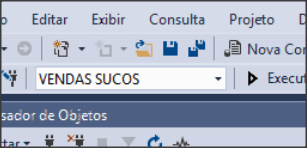




7) Com isso, você terá o banco de dados recuperado com o nome **SUCOS\_VENDAS**. O banco **VENDAS SUCOS** é o banco de dados que você está construindo e o **SUCOS\_VENDAS** fará o papel de banco de dados fonte:



8) Crie uma nova consulta associada à base **VENDAS SUCOS**:



9) Para se referir à tabela que está em outro banco de dados, basta, no nome da tabela, colocar o nome do banco de dados, como mostrado abaixo:

SELECT \* FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

Assim você pode fazer uma seleção na tabela **Tabela de Produtos**, que faz parte do banco fonte.

10) Você pode selecionar os dados da tabela fonte (**Tabela de Produtos**) para incluir na tabela destino (**Produtos**). Mas, para isso, sua seleção na origem tem que resultar em nomes de campos iguais aos contidos na tabela destino:

SELECT [CODIGO DO PRODUTO] AS [CÓDIGO],

[NOME DO PRODUTO] AS [DESCRITOR], [SABOR], [TAMANHO],

[EMBALAGEM], [PREÇO DE LISTA] AS [PREÇO LISTA]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

11) Assim você pode executar o comando abaixo:

INSERT INTO [PRODUTOS] ([CÓDIGO], [DESCRITOR], [SABOR],

[TAMANHO], [EMBALAGEM], [PREÇO LISTA])

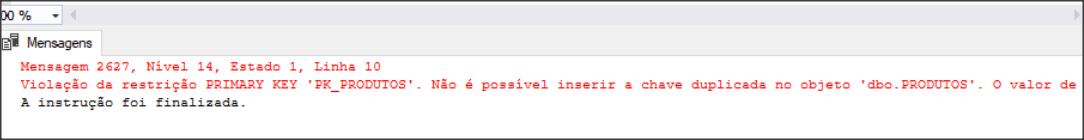
SELECT [CODIGO DO PRODUTO] AS [CÓDIGO],

[NOME DO PRODUTO] AS [DESCRITOR], [SABOR], [TAMANHO],

[EMBALAGEM], [PREÇO DE LISTA] AS [PREÇO LISTA]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

12) A execução deste comando acarretará em um erro de chave primária, pois o produto **1040107** já havia sido inserido na tabela destino em outra aula:



13) Quando um bloco completo é lido e um dos registros deste bloco apresentar um erro, todo o bloco é recusado no momento da gravação. Para isso, retire o produto **1040107** de nossa seleção:

SELECT [CODIGO DO PRODUTO] AS [CÓDIGO],

[NOME DO PRODUTO] AS [DESCRITOR], [SABOR], [TAMANHO],

[EMBALAGEM], [PREÇO DE LISTA] AS [PREÇO LISTA]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

WHERE [CODIGO DO PRODUTO] <> '1040107'

14) Assim, executando o comando abaixo:

INSERT INTO [PRODUTOS] ([CÓDIGO], [DESCRITOR], [SABOR],

[TAMANHO], [EMBALAGEM], [PREÇO LISTA])

SELECT [CODIGO DO PRODUTO] AS [CÓDIGO],

[NOME DO PRODUTO] AS [DESCRITOR], [SABOR], [TAMANHO],

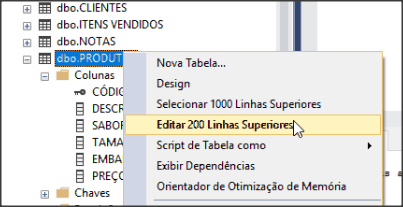
[EMBALAGEM], [PREÇO DE LISTA] AS [PREÇO LISTA]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

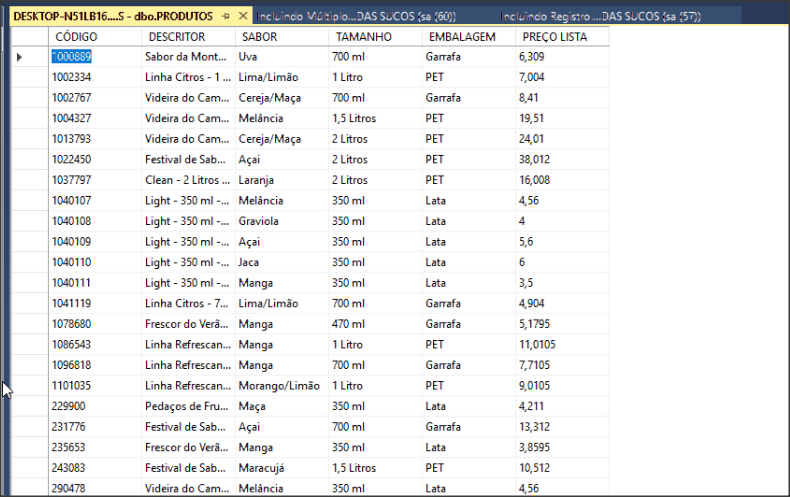
WHERE [CODIGO DO PRODUTO] <> '1040107'

Você terá todos os dados transferidos da tabela fonte para a tabela alvo.

15) Você pode digitar os dados diretamente na tabela. Para isso, clique com o botão da direita do mouse sobre a tabela **PRODUTOS** e selecione a opção **Editar 200 Linhas Superiores**:



16) Você terá:



17) Você pode digitar um registro diretamente na tabela. E os problemas de restrições (chaves primárias ou estrangeiras) serão alertados durante a digitação. Para ver isso, digite a seguinte informação:

CÓDIGO: 838820

DESCRITOR: Clean - 1,5 Litros - Jaca

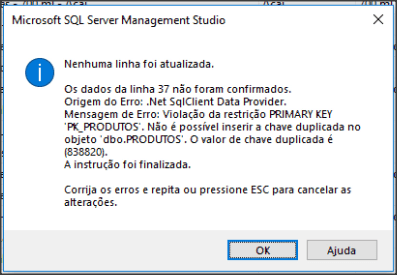
SABOR: Jaca

TAMANHO: 1,5 Litros

EMBALAGEM: PET

PREÇO LISTA: 15,5

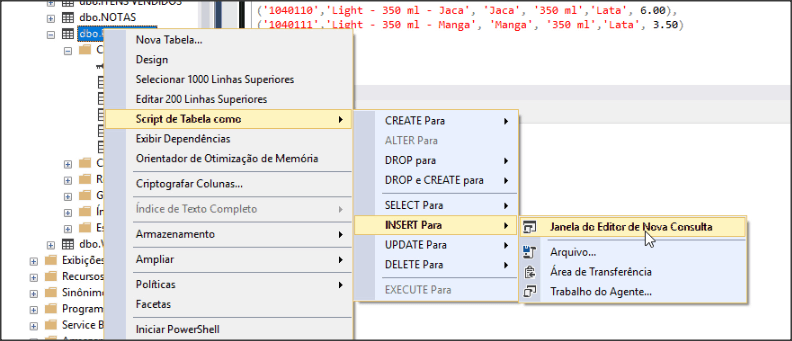
Ao confirmar, você terá o erro:



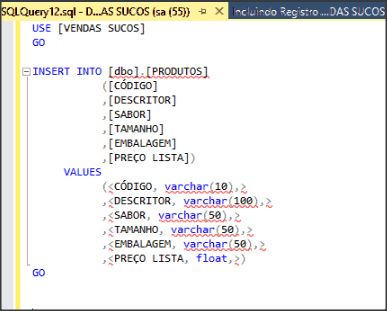
18) Corrija para o código **838821** e inclua o registro:



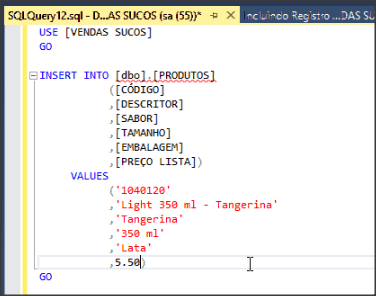
19) Você pode também pedir para que o **SQL Server Management Studio** te ajude a escrever o comando **INSERT** de uma tabela. Para isso, clique com o botão da direita do mouse sobre a tabela **PRODUTOS** e escolha a opção **Script de Tabela como --> INSERT Para --> Janela do Editor de Nova Consulta**:



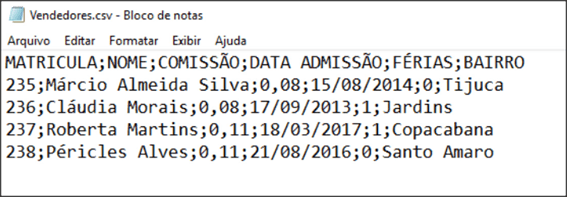
20) Você o comando mostrado abaixo:



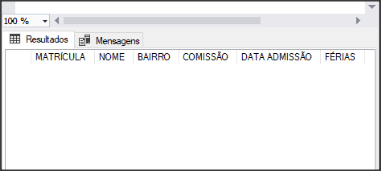
21) Agora basta substituir os dados após **VALUES** para incluir um novo registro na tabela:



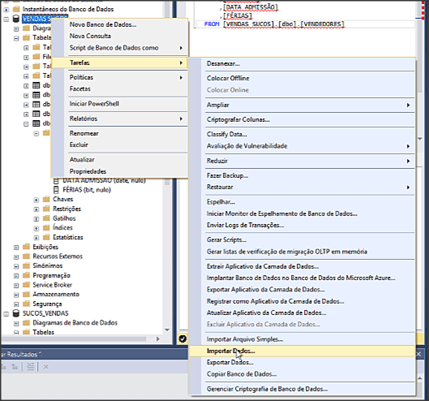
22) Faça o download do arquivo **Vendedores.csv** [aqui](https://s3.amazonaws.com/caelum-online-public/837-manipulacao-dados-com-sql-server-2017/03/Vendedores.csv) e verifique seu conteúdo através de um editor de texto:



23) Volte ao **SQL Server Management Studio** e clique com o botão da direita do mouse sobre a tabela de vendedores, selecione a opção **Selecionar as 1000 Primeiras Linhas Superiores**. Veja que a tabela está vazia:



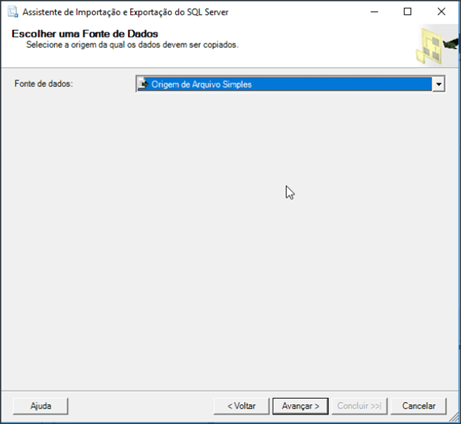
24) Clique com o botão da direita do mouse sobre o banco de dados **VENDAS SUCOS** e escolha a opção **Tarefas --> Importar dados**:



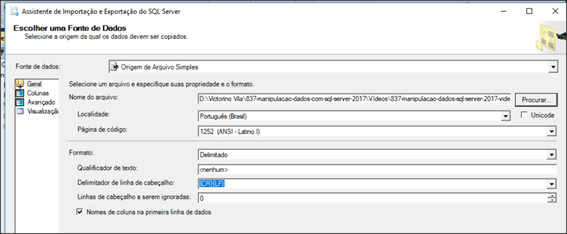
25) Clique em **Avançar**:



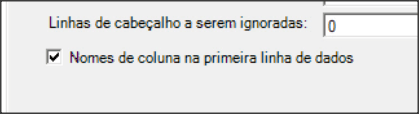
26) Em fonte de dados, selecione **Origem de Arquivo Simples**:



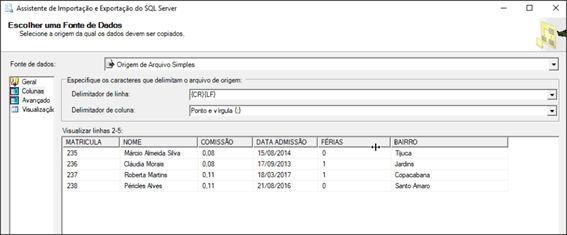
27) Após a escolha da origem de dados, a caixa de diálogo será mostrada como a abaixo. Clique em **Procurar...** e selecione o arquivo que foi baixado na sua máquina (o arquivo **Vendedores.csv**):



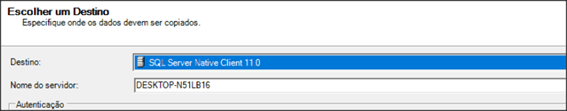
28) Certifique-se que a opção **Nomes de coluna na primeira linha de dados** esteja marcada. Depois clique em **Avançar**:



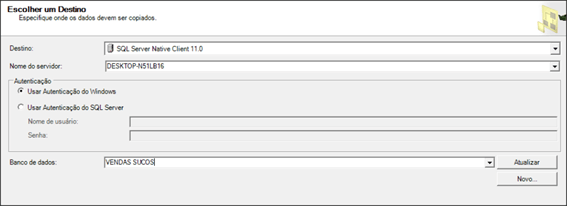
29) Note que você pode visualizar uma amostra dos dados que serão importados do arquivo texto. Depois de visualizá-los, clique em **Avançar**:



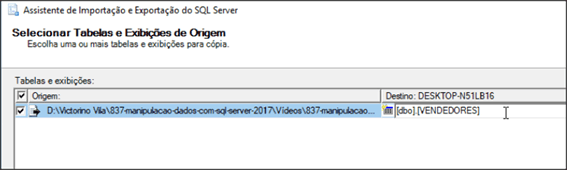
30) Defina o destino, escolhendo **SQL Server Native Client 11.0**:



31) Inclua também as propriedades de conexão na base de dados **VENDAS SUCOS**. Depois clique em **Avançar**:



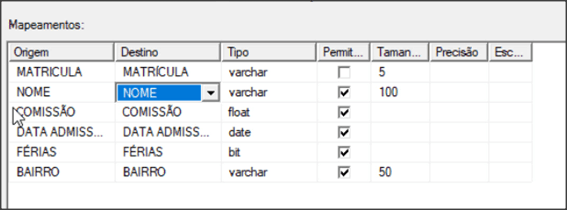
32) Do lado direito, selecione o nome da tabela destino (que será **VENDEDORES**):

.

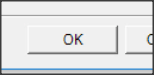
33) Clique no botão **Editar Mapeamentos** para observar se os campos do arquivo de origem estão corretamente associados aos campos da tabela de destino:



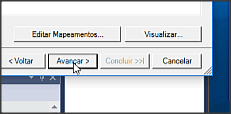
34) A associação será sugerida pelos nomes dos campos. Alguns campos podem não ter esta associação automática configurada. Faça a associação manualmente, conforme mostrado abaixo:



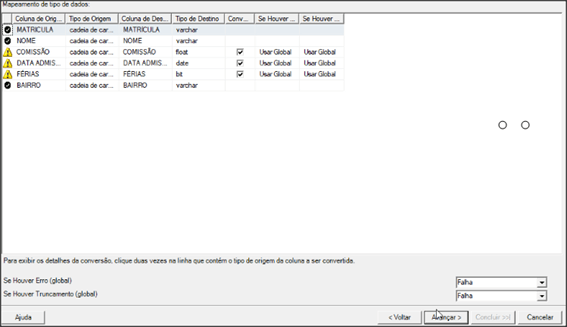
35) Clique em **OK** após confirmar as associações:



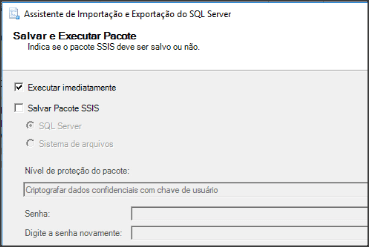
36) Ao voltar na tela onde você associou o arquivo de origem à tabela de destino, clique em **Avançar**:



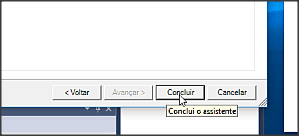
37) Você verá um resumo das associações a serem utilizadas na importação dos dados. Clique novamente em **Avançar**:



38) Veja se a opção **Executar imediatamente** está marcada e clique novamente em **Avançar**:



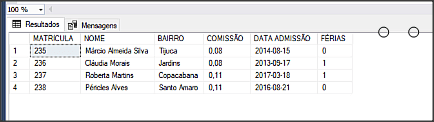
39) Clique em **Concluir**:



40) Veja o processo sendo executado:



41) Clique em **Fechar** e veja novamente o conteúdo da tabela **VENDEDORES**, agora com dados:



1. Observe os valores dos dados referentes ao produto **1040107**. Para tal, crie uma nova consulta associada à base **VENDAS SUCOS**, executando o comando abaixo:

SELECT \* FROM PRODUTOS

Podemos localizá-lo no grid com os resultados:



2) Altere o preço da tabela, executando:

UPDATE [PRODUTOS] SET [PREÇO LISTA] = 5

WHERE [CÓDIGO] = '1040107'

Após a execução, você terá:



1. No mesmo comando **UPDATE**, altere mais de um campo. Digite:

UPDATE [PRODUTOS] SET [SABOR] = 'Laranja', [EMBALAGEM] = 'PET'

WHERE [CÓDIGO] = '1040107'

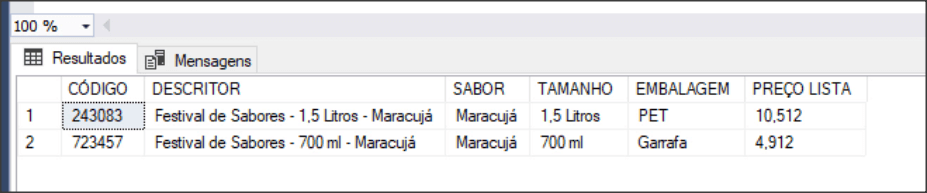
Você terá:



1. O comando **UPDATE** pode ser feito em mais de um registro ao mesmo tempo. Se você executar o comando:

SELECT \* FROM PRODUTOS WHERE [SABOR] = 'Maracujá'

Terá vários produtos com sabor **Maracujá**:



5) Se você executar:

UPDATE [PRODUTOS] SET [PREÇO LISTA] = [PREÇO LISTA] \* 1.10

WHERE [SABOR] = 'Maracujá'

Estará aumentando em 10% os preços de todos os produtos de sabor **Maracujá**.

6) Você pode usar o comando **UPDATE** junto com o **FROM** (como se fosse efetuar um **JOIN** entre tabelas) para alterar dados de uma tabela, baseado no conteúdo de outra. Para isso, mude o valor do preço de um produto na tabela **TABELA DE PRODUTOS**, da base **SUCOS\_VENDAS** que é a tabela fonte, através de uma nova consulta associada à base **VENDAS SUCOS**. Execute:

UPDATE [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

SET [PREÇO DE LISTA] = 7

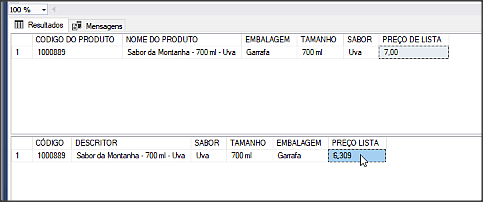
WHERE [CODIGO DO PRODUTO] = '1000889'

7) Se você executar estes dois comandos comparando os preços dos mesmos produtos nas duas tabelas (fonte e alvo), você terá:

SELECT \* FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

WHERE [CODIGO DO PRODUTO] = '1000889'

SELECT \* FROM [PRODUTOS] WHERE [CÓDIGO] = '1000889'



Na tabela origem, o preço é 7 e na destino o preço é 6,309.

8) Sincronize o preço. Para isso, use o comando **UPDATE** com **FROM**. Mas antes execute o comando abaixo:

SELECT B.[CODIGO DO PRODUTO], B.[PREÇO DE LISTA],

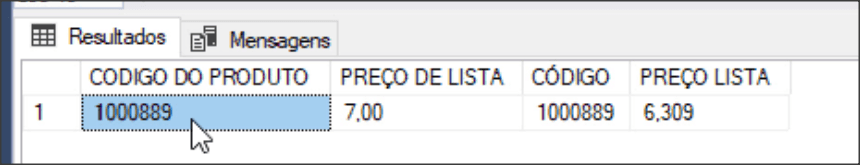
A.[CÓDIGO], A.[PREÇO LISTA] FROM [PRODUTOS] A

INNER JOIN [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

WHERE A.[CÓDIGO] = '1000889'

Este comando faz um **JOIN** entre as duas tabelas (origem e destino) mostrando os dois preços lado a lado:



9) Para haver a sincronização, execute o **UPDATE** usando os mesmos parâmetros do **JOIN** acima:

UPDATE A SET A.[PREÇO LISTA] = B.[PREÇO DE LISTA] FROM [PRODUTOS] A

INNER JOIN [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

WHERE A.[CÓDIGO] = '1000889'

10) Após a execução do comando acima, você pode comparar novamente o preço nas duas tabelas e verá que eles estão sincronizados:

SELECT B.[CODIGO DO PRODUTO], B.[PREÇO DE LISTA],

A.[CÓDIGO], A.[PREÇO LISTA] FROM [PRODUTOS] A

INNER JOIN [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

WHERE A.[CÓDIGO] = '1000889'



11) O comando **MERGE** faz a mesma coisa que o **UPDATE FROM**. Veja: altere o preço de origem de outro produto, executando o comando abaixo:

UPDATE [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS]

SET [PREÇO DE LISTA] = 8

WHERE [CODIGO DO PRODUTO] = '1002334'

12) Comparando este produto na tabela origem e destino, você terá:

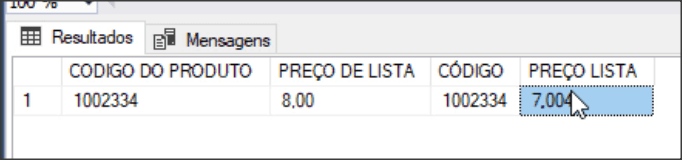
SELECT B.[CODIGO DO PRODUTO], B.[PREÇO DE LISTA],

A.[CÓDIGO], A.[PREÇO LISTA] FROM [PRODUTOS] A

INNER JOIN [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

WHERE [CÓDIGO] = '1002334'



13) Vamos executar o **MERGE**:

MERGE INTO [PRODUTOS] A

USING [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

AND A.[CÓDIGO] = '1002334'

WHEN MATCHED THEN

UPDATE SET A.[PREÇO LISTA] = B.[PREÇO DE LISTA];

14) Após a execução, se você comparar novamente o produto nas duas tabelas, você:

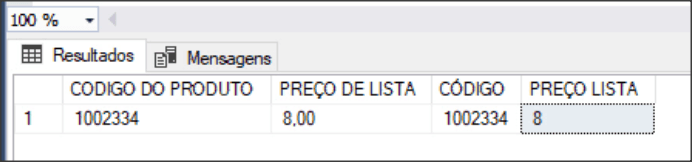
SELECT B.[CODIGO DO PRODUTO], B.[PREÇO DE LISTA],

A.[CÓDIGO], A.[PREÇO LISTA] FROM [PRODUTOS] A

INNER JOIN [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS] B

ON A.[CÓDIGO] = B.[CODIGO DO PRODUTO]

WHERE [CÓDIGO] = '1002334'



15) Baixe o arquivo **Inclusão de registros.sql** [aqui](https://s3.amazonaws.com/caelum-online-public/837-manipulacao-dados-com-sql-server-2017/04/Inclus%C3%A3o+de+registros.sql) e abra-o. Execute os comandos de inclusão de dados conforme mostrados abaixo. Certifique-se que esteja na base **VENDAS SUCOS**:

INSERT INTO [dbo].[PRODUTOS] ([CÓDIGO], [DESCRITOR],

[SABOR], [TAMANHO], [EMBALAGEM], [PREÇO LISTA])

VALUES ('1001001', 'Sabor dos Alpes 700 ml - Manga',

'Manga', '700 ml', 'Garrafa', 7.50),

('1001000', 'Sabor dos Alpes 700 ml - Melão',

'Melão', '700 ml', 'Garrafa', 7.50),

('1001002', 'Sabor dos Alpes 700 ml - Graviola',

'Graviola', '700 ml', 'Garrafa', 7.50),

('1001003', 'Sabor dos Alpes 700 ml - Tangerina',

'Tangerina', '700 ml', 'Garrafa', 7.50),

('1001004', 'Sabor dos Alpes 700 ml - Abacate',

'Abacate', '700 ml', 'Garrafa', 7.50),

('1001005', 'Sabor dos Alpes 700 ml - Açai',

'Açai', '700 ml', 'Garrafa', 7.50),

('1001006', 'Sabor dos Alpes 1 Litro - Manga',

'Manga', '1 Litro', 'Garrafa', 7.50),

('1001007', 'Sabor dos Alpes 1 Litro - Melão',

'Melão', '1 Litro', 'Garrafa', 7.50),

('1001008', 'Sabor dos Alpes 1 Litro - Graviola',

'Graviola', '1 Litro', 'Garrafa', 7.50),

('1001009', 'Sabor dos Alpes 1 Litro - Tangerina',

'Tangerina', '1 Litro','Garrafa',7.50),

('1001010', 'Sabor dos Alpes 1 Litro - Abacate',

'Abacate', '1 Litro','Garrafa',7.50),

('1001011', 'Sabor dos Alpes 1 Litro - Açai',

'Açai','1 Litro','Garrafa',7.50)

16) Execute o comando abaixo para listar todos os produtos com o texto **Sabor dos Alpes**:

SELECT \* from [PRODUTOS]

WHERE SUBSTRING([DESCRITOR],1,15) = 'Sabor dos Alpes'

17) Exclua o registro **1001000**:

DELETE FROM [PRODUTOS] WHERE [CÓDIGO] = '1001000'

18) Selecione todos os produtos com tamanho igual a **1 Litro** e que seja **Sabor dos Alpes**:

SELECT \* FROM [PRODUTOS] WHERE [TAMANHO] = '1 Litro'

AND SUBSTRING([DESCRITOR],1,15) = 'Sabor dos Alpes'



19) Para apagá-los, use o comando **DELETE** com a mesma cláusula **WHERE**:

DELETE FROM [PRODUTOS] WHERE [TAMANHO] = '1 Litro'

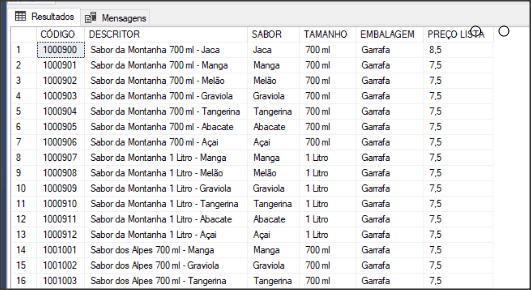
AND SUBSTRING([DESCRITOR],1,15) = 'Sabor dos Alpes'

20) Liste todos os produtos que existam na tabela **PRODUTO**, do banco **VENDAS SUCOS**, e que não existam na tabela **CODIGO DO PRODUTO** do banco de origem (**SUCOS\_VENDAS**):

SELECT \* FROM [PRODUTOS] WHERE [CÓDIGO]

NOT IN (SELECT [CODIGO DO PRODUTO]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS])



Existem 25 linhas.

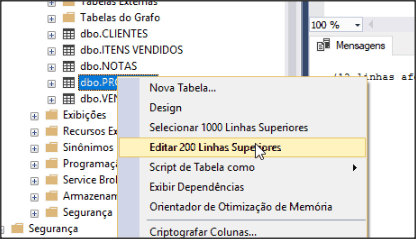
21) Para apagar os registros que existam em **PRODUTO** e não existam em **CODIGO DO PRODUTO**, basta colocar a condição de teste na cláusula **WHERE**:

DELETE FROM [PRODUTOS] WHERE [CÓDIGO]

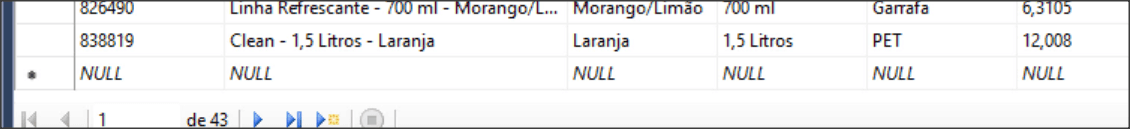
NOT IN (SELECT [CODIGO DO PRODUTO]

FROM [SUCOS\_VENDAS].[DBO].[TABELA DE PRODUTOS])

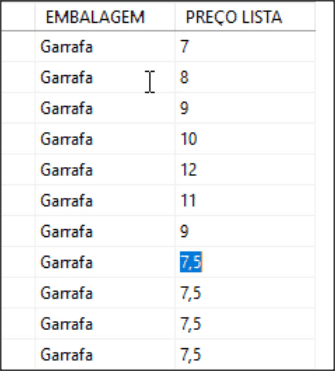
22) Clique com o botão da direita do mouse sobre a tabela **PRODUTOS** e selecione a opção **Editar 200 Linhas Superiores**:



23) Se você for no final da lista, verá o ponto de onde você pode editar novos registros:



24) Faça uma série de edições diretamente no grid para alterar os dados, por exemplo, para alterar os preços:

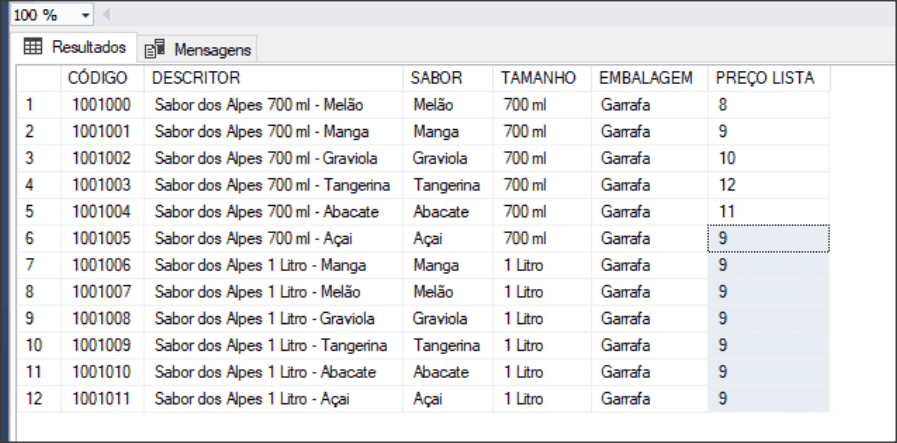


25) A alteração efetuada no grid automaticamente reflete no resultado da tabela. Se você executar:

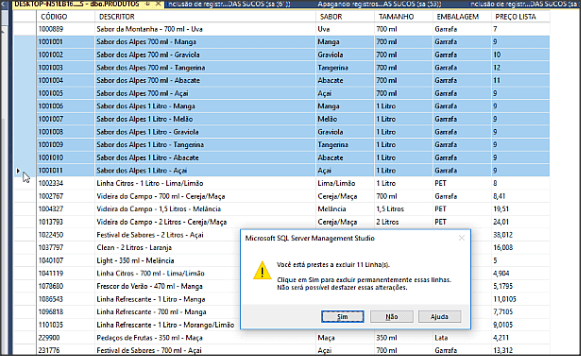
SELECT \* from [PRODUTOS]

WHERE SUBSTRING([DESCRITOR],1,15) = 'Sabor dos Alpes'

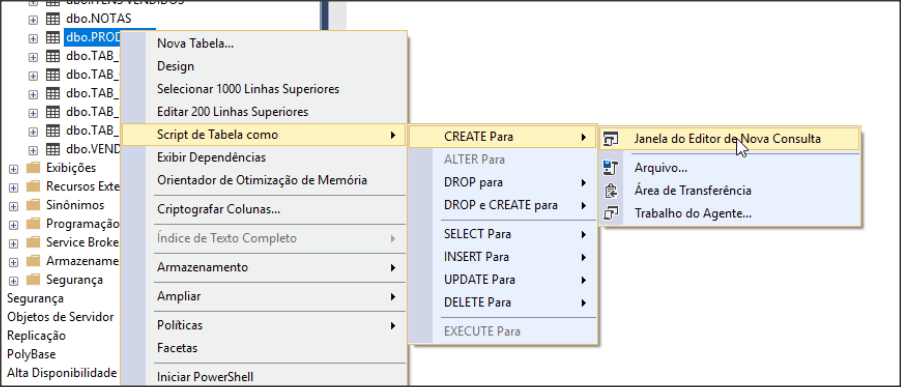
Verá os valores já alterados:



26) Você pode selecionar um bloco no grid e depois exclui-lo teclando **DEL**:



27) Clique com botão da direita do mouse sobre a tabela de produtos e escolha a opção **Script da Tabela como --> CREATE Para --> Janela do Editor de Nova Consulta**:



28) Altere o nome da tabela para **Produtos\_2** e o nome da PK para **PK\_Produtos\_2**. Depois execute a criação da nova tabela:

USE [VENDAS SUCOS]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[PRODUTOS\_2](

[CÓDIGO] [varchar](10) NOT NULL,

[DESCRITOR] [varchar](100) NULL,

[SABOR] [varchar](50) NULL,

[TAMANHO] [varchar](50) NULL,

[EMBALAGEM] [varchar](50) NULL,

[PREÇO LISTA] [float] NULL,

CONSTRAINT [PK\_PRODUTOS\_2] PRIMARY KEY CLUSTERED (

[CÓDIGO] ASC

) WITH (

PAD\_INDEX = OFF,

STATISTICS\_NORECOMPUTE = OFF,

IGNORE\_DUP\_KEY = OFF,

ALLOW\_ROW\_LOCKS = ON,

ALLOW\_PAGE\_LOCKS = ON

) ON [PRIMARY]) ON [PRIMARY]

GO

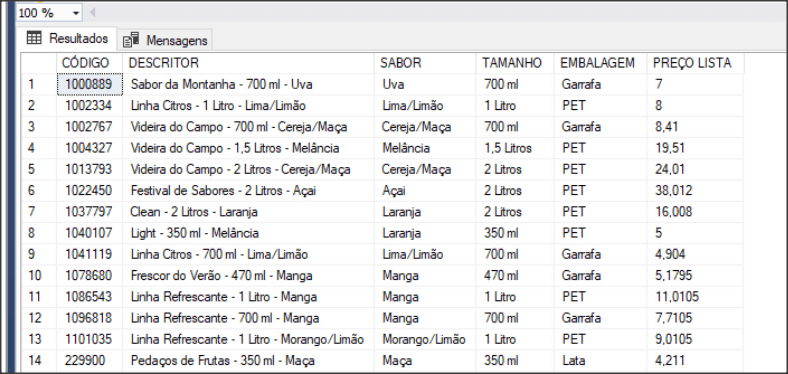
29) Inclua os dados contidos na tabela original (**Produtos**) para esta nova, executando o comando abaixo:

INSERT INTO PRODUTOS\_2

SELECT \* FROM PRODUTOS

30) Agora você tem produtos na tabela nova:

SELECT \* FROM PRODUTOS\_2



31) Você pode fazer uma alteração em todos os registros da tabela. No exemplo abaixo, aumente o preço de todos os produtos em 10%:

UPDATE PRODUTOS\_2

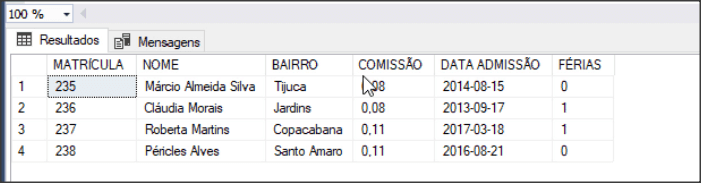
SET [PREÇO LISTA] = [PREÇO LISTA] \* 1.10

32) Você também pode apagar todos os registros da tabela, executando:

DELETE FROM PRODUTOS\_2

33) Verifique o conteúdo da tabela de vendedores, executando:

SELECT \* FROM [VENDEDORES]



34) Execute o comando abaixo para iniciar uma nova transação:

BEGIN TRANSACTION

35) Efetue uma alteração da comissão dos vendedores, aumentando-a em 15%:

UPDATE [VENDEDORES] SET [COMISSÃO] = [COMISSÃO] \* 1.15

36) Inclua um novo vendedor:

INSERT INTO [VENDEDORES] ([MATRÍCULA], [NOME], [BAIRRO],

[COMISSÃO], [DATA ADMISSÃO], [FÉRIAS])

VALUES ('99999','João da Silva','Icaraí',0.08,'2014-09-01',0)

37) Agora, se você verificar a tabela de vendedores, terá:

SELECT \* FROM [VENDEDORES]



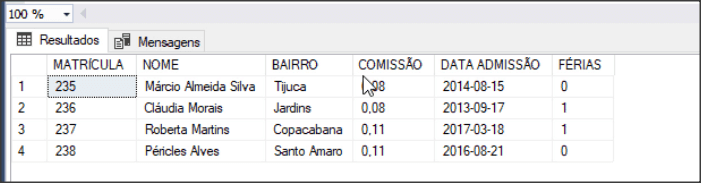
38) Agora, executando o comando abaixo:

ROLLBACK

Você volta ao status antes do **BEGIN TRANSACTION**.

39) Verifique agora a tabela de vendedores:

SELECT \* FROM [VENDEDORES]



40) Se você fizer novamente o início de uma nova transação:

BEGIN TRANSACTION

41) E executar novamente a alteração de comissão e inclusão de novo vendedor:

UPDATE [VENDEDORES] SET [COMISSÃO] = [COMISSÃO] \* 1.15

INSERT INTO [VENDEDORES] ([MATRÍCULA], [NOME], [BAIRRO],

[COMISSÃO], [DATA ADMISSÃO], [FÉRIAS])

VALUES ('99999','João da Silva','Icaraí',0.08,'2014-09-01',0)

42) Ao executar o comando abaixo, as alterações são confirmadas:

COMMIT

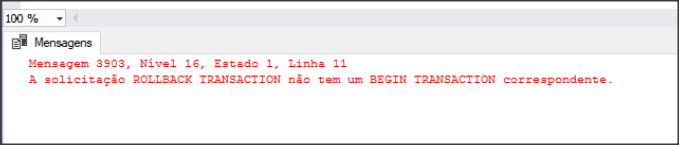
43) Verificando agora a tabela:

SELECT \* FROM [VENDEDORES]

Você terá:



44) Se você executar o comando de **ROLLBACK**, terá um erro, porque o comando **COMMIT** encerrou a última transação:



1) Crie uma tabela com o campo auto-incremento, conforme mostrado abaixo:

CREATE TABLE TAB\_IDENTITY (

ID INT IDENTITY (1,1) NOT NULL,

DESCRITOR VARCHAR(20) NULL

)

2) Insira alguns registros, note que nos comandos de **INSERT** abaixo, não foi especificado o valor para o campo de auto-incremento:

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE X')

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE Y')

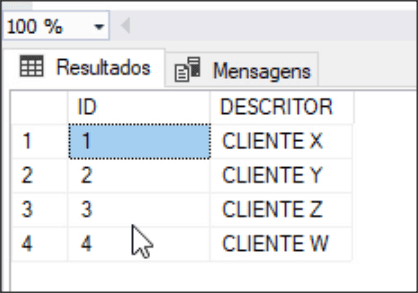
INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE Z')

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE W')

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE A')

3) Se você selecionar dados da tabela, verá o valor do campo auto-incremento que foi incluído automaticamente pelo SQL Server:

SELECT \* FROM TAB\_IDENTITY



4) Exclua um dos registros da tabela acima:

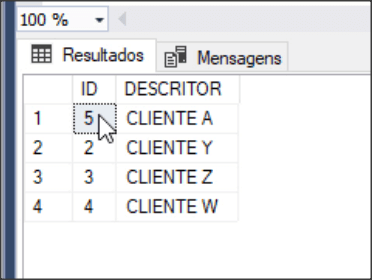
DELETE FROM TAB\_IDENTITY WHERE ID = 1

5) Se você incluir um novo registro:

INSERT INTO TAB\_IDENTITY (DESCRITOR) VALUES ('CLIENTE A')

6) O auto-incremento continuará a crescer seu valor:

SELECT \* FROM TAB\_IDENTITY



7) Você pode criar um campo de auto-incremento com valor inicial e intervalo de crescimento diferente de **(1,1)**. Para isso, elimine a tabela:

DROP TABLE TAB\_IDENTITY

8) Crie novamente a tabela, mas agora o campo auto-incremento com as propriedades **(100,5)**:

CREATE TABLE TAB\_IDENTITY (

ID INT IDENTITY (100,5) NOT NULL,

DESCRITOR VARCHAR(20) NULL

)

9) Inclua novamente os registros:

INSERT INTO TAB\_IDENTITY (DESCRITOR)

VALUES ('CLIENTE X')

INSERT INTO TAB\_IDENTITY (DESCRITOR)

VALUES ('CLIENTE Y')

INSERT INTO TAB\_IDENTITY (DESCRITOR)

VALUES ('CLIENTE Z')

INSERT INTO TAB\_IDENTITY (DESCRITOR)

VALUES ('CLIENTE W')

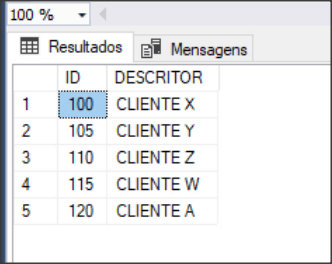
INSERT INTO TAB\_IDENTITY (DESCRITOR)

VALUES ('CLIENTE A')

10) Selecionando os dados:

SELECT \* FROM TAB\_IDENTITY

Você terá:



11) Crie uma nova consulta no Management Studio e digite os comandos abaixo para criar uma nova tabela. Execute o comando:

CREATE TABLE TAB\_PADRAO (

ID INT IDENTITY (1,1) NOT NULL,

DESCRITOR VARCHAR(20) NULL,

ENDERECO VARCHAR(200) NULL,

CIDADE VARCHAR(50) DEFAULT 'Cidade não definida',

DATA\_CRIACAO DATE DEFAULT GETDATE()

)

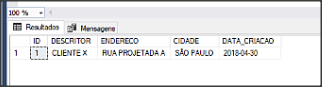
12) Agora digite um novo comando de inclusão de registros:

INSERT INTO TAB\_PADRAO (DESCRITOR, ENDERECO, CIDADE, DATA\_CRIACAO)

VALUES ('CLIENTE X', 'RUA PROJETADA A', 'SÃO PAULO', '2018-04-30')

13) Fazendo a seleção dos registros, você terá:

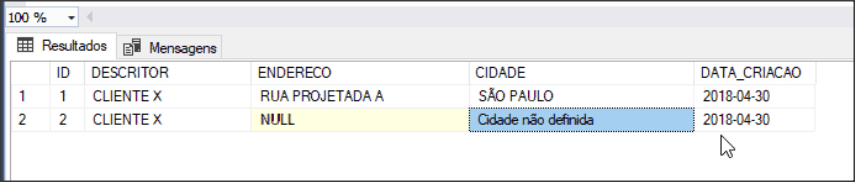
SELECT \* FROM TAB\_PADRAO



14) Agora, inclua novo registro, omitindo uma série de valores:

INSERT INTO TAB\_PADRAO (DESCRITOR) VALUES ('CLIENTE X')

15) Se você observar a tabela, verá uma série de valores nulos para os campos omitidos, exceto no campo **CIDADE**, que possui um valor padrão definido na criação da tabela:



16) Crie uma nova consulta no Management Studio e digite os comandos abaixo para criar uma nova tabela::

CREATE TABLE TAB\_FATURAMENTO (

DATA\_VENDA DATE NULL,

TOTAL\_VENDA FLOAT

)

17) Agora, crie uma **TRIGGER** que irá apagar todo o conteúdo da tabela criada anteriormente e calcular o faturamento total de todas as notas fiscais por dia:

CREATE TRIGGER TG\_ITENS\_VENDIDOS

ON [ITENS VENDIDOS]

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

DELETE FROM TAB\_FATURAMENTO;

INSERT INTO TAB\_FATURAMENTO (DATA\_VENDA, TOTAL\_VENDA)

SELECT A.DATA AS DATA\_VENDA,

SUM(B.QUANTIDADE \* B.[PREÇO]) AS TOTAL\_VENDA

FROM NOTAS A INNER JOIN [ITENS VENDIDOS] B

ON A.NÚMERO = B.NÚMERO

GROUP BY A.DATA;

END;

18) insira uma nota fiscal e alguns itens:

INSERT INTO [NOTAS] ([NÚMERO], DATA, CPF, MATRICULA, IMPOSTO)

VALUES ('0100', '2018-05-15', '1471156710', '235', 0.18)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

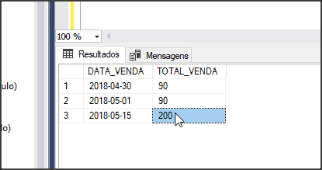
VALUES ('0100', '1000889', 100, 1)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

VALUES ('0100', '1002334', 100, 1)

19) Após a inclusão desta nota fiscal, verifique a tabela com os faturamentos consolidados:

SELECT \* FROM TAB\_FATURAMENTO



20) Insira uma nova nota fiscal com seus respectivos itens:

INSERT INTO [NOTAS] ([NÚMERO], DATA, CPF, MATRICULA, IMPOSTO)

VALUES ('0101', '2018-05-15', '1471156710', '235', 0.18)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

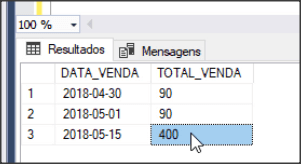
VALUES ('0101', '1000889', 100, 1)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

VALUES ('0101', '1002334', 100, 1)

21) Verificando a tabela de faturamentos consolidados, você verá novos valores:

SELECT \* FROM TAB\_FATURAMENTO



22) Mais uma nota, agora para o dia 16, e você terá novos valores:

INSERT INTO [NOTAS] ([NÚMERO], DATA, CPF, MATRICULA, IMPOSTO)

VALUES ('0102', '2018-05-16', '1471156710', '235', 0.18)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

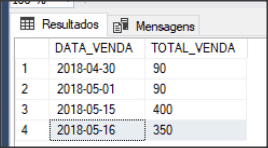
VALUES ('0102', '1000889', 100, 1.5)

INSERT INTO [ITENS VENDIDOS] ([NÚMERO],[CÓDIGO],[QUANTIDADE],[PREÇO])

VALUES ('0102', '1002334', 200, 1)

Depois:

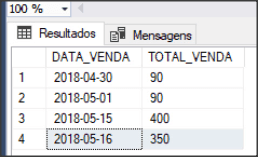
SELECT \* FROM TAB\_FATURAMENTO



23) Qualquer alteração na tabela irá mudar a tabela de faturamentos, já que você colocou a propriedade **AFTER INSERT, UPDATE, DELETE**. Se você executar o comando de exclusão de um item de nota:

DELETE FROM [ITENS VENDIDOS] WHERE [NÚMERO] = '0102' AND [CÓDIGO] = '1002334'

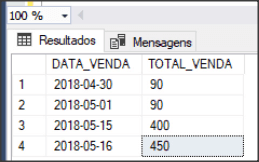
Terá novos valores na tabela de faturamento consolidado:



24) Também haverá mudanças ao executar um **UPDATE**:

UPDATE [ITENS VENDIDOS] SET [QUANTIDADE] = 300

WHERE [NÚMERO] = '0102' AND [CÓDIGO] = '1000889'



25) Crie uma nova consulta no Management Studio e digite os comandos abaixo para criar uma nova tabela:

CREATE TABLE TAB\_CHECK (

ID INT NOT NULL,

NOME VARCHAR(20) NULL,

IDADE INT NULL,

CIDADE VARCHAR(20) NULL,

CONSTRAINT CHK\_PESSOA CHECK (IDADE >= 18)

)

26) Na tabela acima, a idade foi restringida para maior de 18 anos. Se você executar o comando:

INSERT INTO TAB\_CHECK (ID, NOME, IDADE, CIDADE)

VALUES (1, 'JOÃO', 19, 'RIO DE JANEIRO')

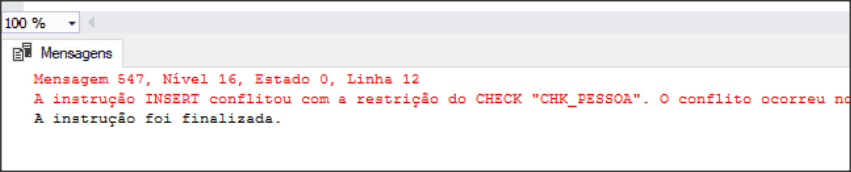
Ele será incluído na tabela.

27) Mas se você executar outro **INSERT**, conforme mostrado abaixo:

INSERT INTO TAB\_CHECK (ID, NOME, IDADE, CIDADE)

VALUES (2,'PEDRO', 17, 'SÃO PAULO')

Terá a mensagem:



28) Se corrigir para:

INSERT INTO TAB\_CHECK (ID, NOME, IDADE, CIDADE)

VALUES (2,'PEDRO', 20, 'SÃO PAULO')

Agora sim a informação é incluída.

29) A condição do **CHECK** pode ser composta com **AND** ou **OR**, por exemplo. Crie outra tabela:

CREATE TABLE TAB\_CHECK2 (

ID INT NOT NULL,

NOME VARCHAR(20) NULL,

IDADE INT NULL,

CIDADE VARCHAR(20) NULL,

CONSTRAINT CHK\_PESSOA2

CHECK (IDADE >= 18 AND CIDADE = 'RIO DE JANEIRO')

)

30) Ao incluir um novo registro:

INSERT INTO TAB\_CHECK2 (ID, NOME, IDADE, CIDADE)

VALUES (1, 'JOÃO', 19, 'RIO DE JANEIRO')

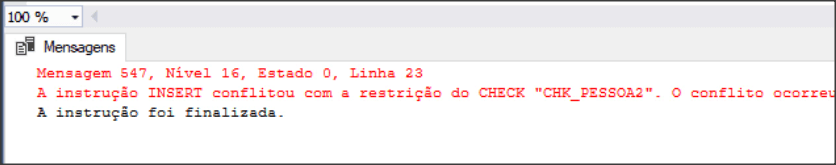
Não haverá problemas.

31) Mas se você digitar:

INSERT INTO TAB\_CHECK2 (ID, NOME, IDADE, CIDADE)

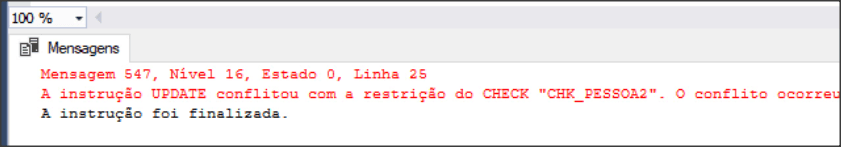
VALUES (2, 'PEDRO', 20, 'RIO DE JANEIRO')

O problema de integridade será apresentado, já que a idade está correta, mas a cidade não:



32) A restrição é testada não somente durante o **INSERT**, mas também durante o **UPDATE**:

UPDATE TAB\_CHECK2 SET CIDADE = 'SÃO PAULO' WHERE ID = 2



[00:00] Já sabemos criar banco de dados. O próximo passo é criar tabelas. Como vimos na definição de "tabelas", que foi colocada no início do curso, sabemos que para definirmos uma tabela, precisamos basicamente dizer quantos campos essa tabela vai ter e qual é o tipo de cada campo.

[00:25] Um dado dentro de um campo, não pode ter tipos diferentes. Então, óbvio, para criar tabela, a primeira coisa que temos que aprender é: quais são os tipos disponíveis para usarmos na criação da tabela no SQL Server?

[00:44] Nós temos vários tipos, que são organizados em grupos. O primeiro grupo que eu estou mostrando aqui em cima, são os numéricos exatos. Ele tem esse nome "numéricos exatos", porque eu defino muito bem o tamanho e o número de casas decimais que esses números vão ter.

[01:04] A gente começa pelo Bigint. O Bigint, como o próprio nome em inglês diz, é um inteiro grande, ou seja, é um número que não tem casa decimal e possui um limite, inferior e superior, que está sendo mostrado aqui em cima. Ele vai de -2^63 a +2^63-1. Então, é assim o limite do Bigint.

[01:34] O Numeric, é um número com casas decimais, porém, com casas decimais fixas. Se eu digo que um Numeric tem cinco decimais e se eu representar, por exemplo o número 1 dentro do banco de dados, ele vai ser representado com 1,00000, ou seja, cinco casas decimais. Independente do número ser inteiro ou possuir casas decimais realmente.

[02:02] No Numeric, eu posso definir números de até 38 casas decimais. O tamanho desse campo vai variar pelo número de casas decimais que eu definir.

[02:14] O Bit, é um inteiro que me dá o valor 1 ou 0. Ele é usado normalmente para definir campos lógicos. Normalmente utilizamos o 1 como verdadeiro e o 0 como falso, ou vice-versa, não lembro direito, mas acho que é isso mesmo: o 1 é verdadeiro e o 0, falso. Ou seja, é um campo que dá um status por exemplo, do que ele significa.

[02:44] O Smallint, é um inteiro menor que tem as mesmas definições do Bigint e cujo os limites são um pouco menores do que o Bigint. Na verdade, são bem menores, praticamente quatro vezes menores.

[03:00] O Decimal é igual ao Numeric em sua definição. Tanto faz você definir o campo como decimal ou numeric, eles são a mesma coisa: números com casas decimais fixas.

[03:12] Smallmoney. Normalmente um campo money representa valores financeiros, e além de ter casas decimais, esse valores financeiros são representados dentro do banco de dados com um prefixo, que é o símbolo da moeda que ele representa. Pode ser o símbolo do dólar, do real, do euro, e dependendo do idioma ou da propriedade específica do idioma em que o SQL Server está rodando, você já vai ter automaticamente esses símbolos.

[03:50] Claro que ele é Smallmoney, porque o limite de valores dele é menor, vai de (-214.000 a +214.000 e uns quebrados).

[04:03] O Int, é o inteiro cujo o tamanho está entre o Bit e o Smallint. Ele vai de -2^31 a +2^31-1.

[04:15] O Tinyint, é um inteiro bem pequeno, eu o represento de 0 a 255.

[04:23] O Money, já é a representação de valores financeiros um pouco maior. Na verdade, muito maior, vai de -922.337.203.685.477. Então, o menor valor e o maior valor, é o mesmo menos um.

[04:53] Claro, o Money também tem o símbolo da moeda representado na frente. Esses são os numéricos exatos. Partiremos para o segundo grupo: os numéricos aproximados.

[05:04] Eles têm esse nome, porque na verdade, fazem um arredondamento, são o que chamamos de "números com ponto flutuante". Nós temos dois tipos: o Float e o Real.

[05:19] O Float é o número que tem um range de tamanho, muito maior, ou seja, a sua representação é muito maior. Enquanto o Real, é esse número com um limite um pouco menor.

[05:37] Data e hora: Os campos do tipo data e hora, representam algum período do tempo. Então, nós temos o Data, que me dá o dia, mês e ano. Na verdade, normalmente eu represento no SQL Server ano, mês e dia.

[05:55] Depois, nós vamos ver em outros tópicos do curso, que nós podemos por exemplo, exibir o resultado em determinados formatos diferentes.

[06:05] O DateTimeOffset, é o ano/mês/dia, com hora/minuto/segundo, porém, com a informação do fuso horário. Dependendo de como estou manipulando o meu dado, vou ter além da hora, qual é o deslocamento em relação ao ponto de Greenwich que aquele horário representa.

[06:32] Eu acho que aqui, por exemplo, Rio de Janeiro, São Paulo ou sudeste do Brasil, se eu não me engano… Eu posso estar falando besteira, porque eu não conferi essa informação antes, mas acho que é -4 em relação à Greenwich.

[06:49] DateTime2: eu vejo a data, porém só com hora e minuto, o segundo sempre vai vir 00.

[07:02] O Smalldate, é a data com a hora apenas, não tem minutos e não tem segundos.

[07:10] O Datetime, nessa sim tenho a data, com a hora, minuto e segundo, porém, não tem a representação do fuso horário. Quem tem isso é o DateTimeOffset.

[07:22] E com o Time, eu só vejo a representação da hora.

[07:31] Cadeia de caracteres: nós temos dois grupos, as unicodes e as não unicodes. Já vou explicar o que significa unicode caso você não saiba. Primeiro, eu tenho o Char, o Varchar e o Texto. Os três são cadeias de caracteres não unicode. Sendo que o Char tem o valor fixo de 1 a 8000. O Varchar, também.

[07:58] A diferença do Char e do Varchar, é a seguinte: o Varchar tem o tamanho variável, ou seja, se eu digo que o meu Varchar é tamanho 10, mas eu salvo no campo uma palavra com dois caracteres, ele só vai armazenar aqueles dois caracteres.

[08:17] Se o campo é um Char de 10, e eu gravo o mesmo campo com dois caracteres, ele vai ter que salvar dois caracteres e oito caracteres vazios, ou seja, ele vai ocupar os dez caracteres. Então, a diferença do Char para o Varchar, é que o Char é fixo, e o Varchar é variável.

[08:38] Tem uma outra coisa. O Varchar tem um parâmetro chamado Max, que é para eu representar caracteres muito grandes. Se o campo for um Varchar de Max, ele pode receber até dois gigabytes de tamanho.

[08:58] E o campo Texto, é uma cadeia de caracteres não unicode, mas que representa dados maiores do que o Varchar pode armazenar, ou dados binários. Bem, nChar, nVarchar e nTexto, têm as mesmas definições que o Char, o Varchar e Texto, porém, agora, são caracteres unicode.

[09:26] Bem, é aquela pergunta: o que significa "unicode"? Unicode significa uma cadeia de caracteres que fornece num único conjunto de dados, todos os caracteres de todos os idiomas do mundo.

[09:42] Nós, ocidentais, português e línguas latinas, por exemplo, usamos o abecedário normal, que vai de A a Z. Porém, existem outras línguas como o russo, grego, hebraico, árabe. As línguas orientais, como por exemplo, chinês, coreano, japonês, que utilizam uma cadeia de caracteres diferentes, são outros símbolos, outras formas de representar as letras.

[10:12] Então, se eu vou armazenar no meu banco de dados uma informação que possui caracteres diferentes dos normais, tenho que usar o nVarchar ou nChar ou nTexto, porque ele vai acertar cadeias de caracteres unicode.

[10:33] Finalmente, caracteres binários, que representam dados de, por exemplo: eu tenho lá no meu diretório um arquivo-texto salvo. Eu posso pegar aquele arquivo-texto e salvar num campo do meu SQL Server. Ele será salvo como "binário".

[10:55] A mesma coisa vai acontecer com o VarBinário. A diferença, está no tamanho. O Binário armazena até 8.000 caracteres de tamanho, enquanto o VarBinário, chega até q 2^31-1 caracteres.

Ex:

CREATE TABLE [TABELA DE CLIENTES]

(

[CPF] [VARCHAR] (11) ,

[NOME] [VARCHAR] (100) ,

[ENDERECO 1] [VARCHAR] (150) ,

[ENDERECO 2] [VARCHAR] (150) ,

[BAIRRO] [VARCHAR] (50) ,

[CIDADE] [VARCHAR] (50) ,

[ESTADO] [VARCHAR] (2) ,

[CEP] [VARCHAR] (8) ,

[DATA DE NASCIMENTO] [DATE],

[IDADE] [SMALLINT],

[SEXO] [VARCHAR] (1) ,

[LIMITE DE CREDITO] [MONEY] ,

[VOLUME DE COMPRA] [FLOAT] ,

[PRIMEIRA COMPRA] [BIT]

)

**Criando a primeira tabela**

1) Acesse o **Management Studio**.

2) Crie a tabela de cliente, digitando o comando abaixo:

CREATE TABLE [TABELA DE CLIENTES](

[CPF] [VARCHAR] (11) ,

[NOME] [VARCHAR] (100) ,

[ENDERECO 1] [VARCHAR] (150) ,

[ENDERECO 2] [VARCHAR] (150) ,

[BAIRRO] [VARCHAR] (50) ,

[CIDADE] [VARCHAR] (50) ,

[ESTADO] [VARCHAR] (2) ,

[CEP] [VARCHAR] (8) ,

[DATA DE NASCIMENTO] [DATE],

[IDADE] [SMALLINT],

[SEXO] [VARCHAR] (1) ,

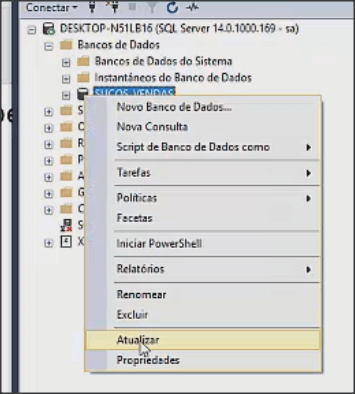
[LIMITE DE CREDITO] [MONEY] ,

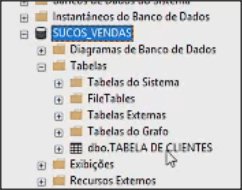
[VOLUME DE COMPRA] [FLOAT] ,

[PRIMEIRA COMPRA] [BIT]

)

3) Execute o comando e depois atualize a árvore de objetos do SQL Server para observar a nova tabela criada:





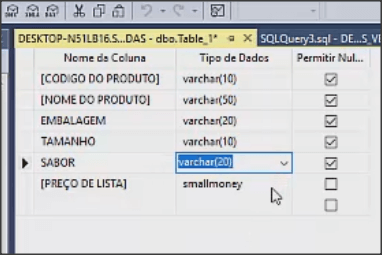
**Criando uma tabela através de assistente**

4) Crie agora uma outra tabela, porém usando o **Assistente do Management Studio**. Clique com o botão da direita do mouse sobre o nome do banco de dados e selecione **Novo --> Tabela**:

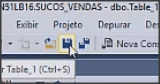


5) No *grid*, preencha com os dados da nova tabela, conforme mostrado abaixo:

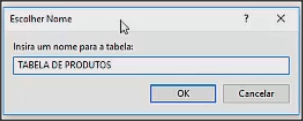
* **[CODIGO DO PRODUTO] Varchar(10)**
* **[NOME DO PRODUTO] Varchar(50)**
* **[EMBALAGEM] Varchar(20)**
* **[TAMANHO] Varchar(10)**
* **[SABOR] Varchar(20)**
* **[PREÇO DE LISTA] Smallmoney**



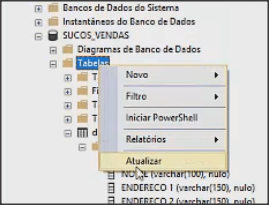
6) Clique no botão **Salvar**:

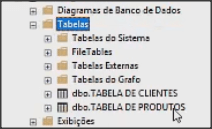


7) Inclua o nome da tabela:

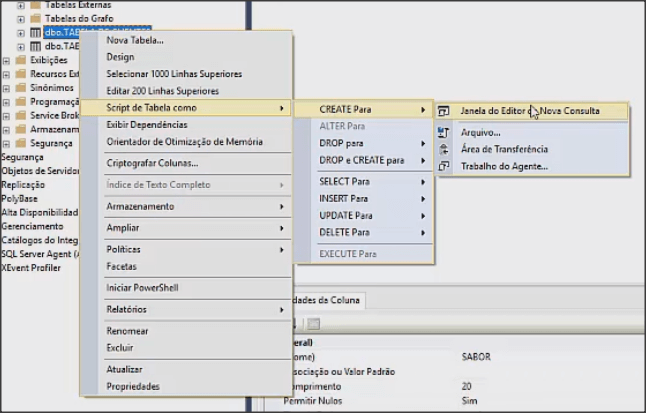


8) Clique em **OK**, depois clique com o botão da direita do mouse sobre **Tabelas** e selecione a opção **Atualizar**. Você verá a nova tabela no banco de dados:

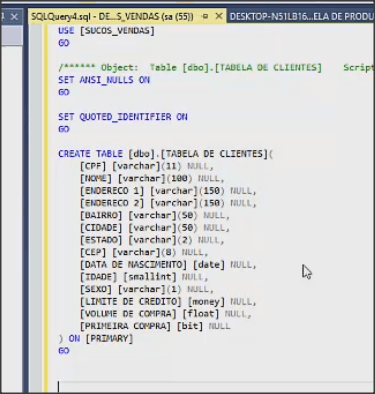




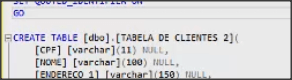
9) Clique com o botão da direita do mouse sobre a tabela de clientes e selecione **Script de Tabela como --> CREATE Para --> Janela do Editor da Nova Consulta**:



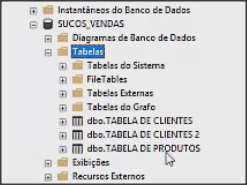
10) Você terá:



11) Modifique o nome da tabela no script para **TABELA DE CLIENTES 2**:

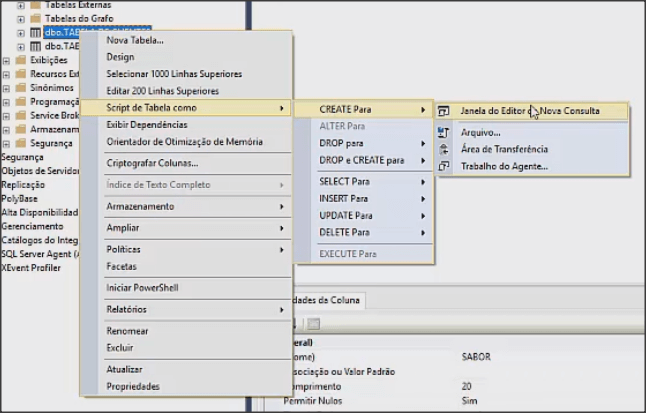


12) Execute. Depois de atualizar os objetos você terá a nova tabela listada:

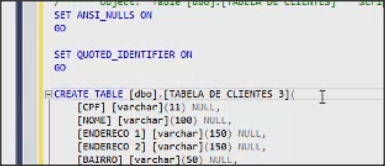


**Apagando uma tabela**

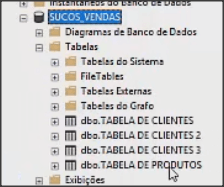
13) Clique com o botão da direita do mouse sobre **Tabela de Clientes** e selecione **Script de Tabela como --> CREATE Para --> Janela do Editor da Nova Consulta**:



14) Edite o nome da tabela para **Tabela de Clientes 3**:



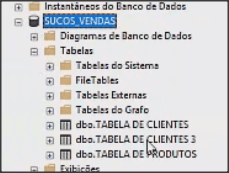
15) Execute, atualize e verifique se a tabela foi criada:



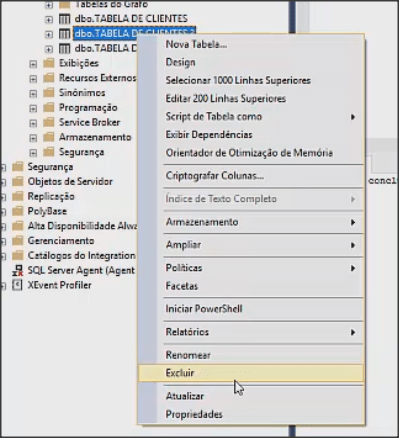
16) Apague a **Tabela de Clientes 2**. Para isso, crie uma nova consulta e digite o comando:

DROP TABLE [dbo].[TABELA DE CLIENTES 2]

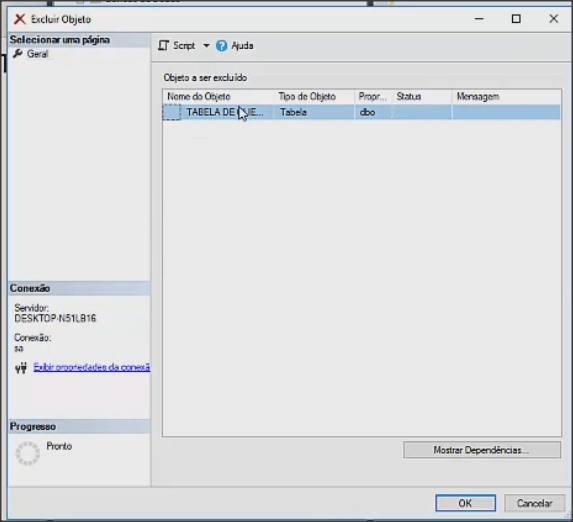
17) Execute o comando, atualize as tabelas e verifique que a tabela não existe mais:



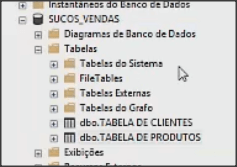
18) Exclua a **Tabela de Clientes 3** através do assistente. Para isso, clique com o botão da direita do mouse sobre ela e escolha a opção **Excluir**:



19) Você pode ver a caixa de diálogo abaixo:



20) Clique em **OK**. A tabela foi excluída:



use [SUCOS\_VENDAS]

INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PRECO DE LISTA])

VALUES

('1040107','Light - 350 ml - Melancia', 'Lata', '350 ml', 'Melancia', 4.56),

('1037797','Clean - 2 Litros - Laranja', 'PET', '2 - Litros', 'Laranja', 16.01),

('1000889','Sabor da Montanha - 700 ml - Uva','Garrafa','700 ml','Uva',6.31),

('1004327','Videira do Campo - 1,5 Litros - Melancia', 'PET', '1,5 Litros','Melancia',19.51)

select \* from [TABELA DE PRODUTOS]

UPDATE [TABELA DE PRODUTOS] SET

[EMBALAGEM] = 'Lata',

[PREÇO DE LISTA] = 2.46

WHERE [CODIGO DO PRODUTO] = '544931'

UPDATE [TABELA DE PRODUTOS] SET

[EMBALAGEM] = 'Garrafa'

WHERE [CODIGO DO PRODUTO] = '1078680'

DELETE FROM [TABELA DE PRODUTOS]

WHERE [CODIGO DO PRODUTO] = '1088126'

ALTER TABLE [TABELA DE PRODUTOS]

ADD CONSTRAINT PK\_PRODUTOS

PRIMARY KEY CLUSTERED ([CODIGO DO PRODUTO])

ALTER TABLE [TABELA DE PRODUTOS]

ALTER COLUMN [CODIGO DO PRODUTO]

VARCHAR(10) NOT NULL

Dica para inclusão de data ( ano-mês-dia ) >> '2021-11-02'

**Incluindo dados na tabela**

1) Acesse o **Management Studio**.

2) Crie uma nova consulta e digite o comando abaixo:

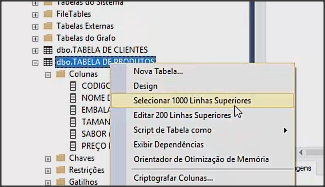
INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

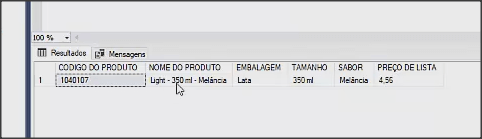
VALUES

('1040107','Light - 350 ml - Melancia', 'Lata', '350 ml', 'Melancia', 4.56)

3) Clique com o botão da direita do mouse sobre a tabela **Tabela de Produtos** e escolha a opção **Selecionar 1000 Linhas Superiores**:



4) Você verá que o novo registro foi incluído na tabela:



**Incluindo registros ao mesmo tempo**

5) Você pode incluir vários registros. Crie uma nova consulta e digite:

INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

('1037797','Clean - 2 Litros - Laranja', 'PET', '2 Litros', 'Laranja', 16.01)

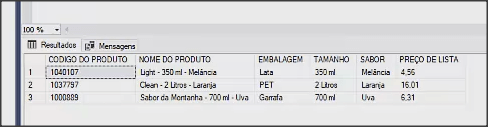
INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

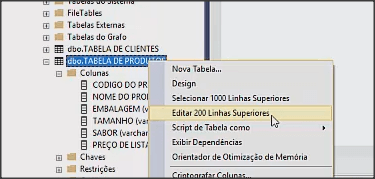
('1000889','Sabor da Montanha - 700 ml - Uva', 'Garrafa', '700 ml', 'Uva', 6.31)

6) Execute o comando e verifique o conteúdo da tabela, escolhendo a opção **Selecionar 1000 Linhas Superiores**:



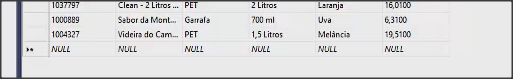
**Incluindo um registro direto na tabela**

7) Clique com o botão da direita do mouse sobre a tabela **Tabela de Produtos** e selecione **Editar 200 Linhas Superiores**:



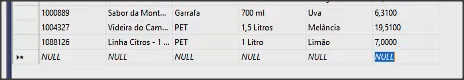
8) Digite o novo campo no *grid*:

* **1004327**
* **Videira do Campo 1,5 Litros Melancia**
* **PET**
* **1,5 Litros**
* **Melancia**
* **19,51**

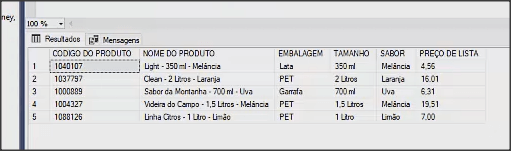


9) Inclua mais uma linha com dados:

* **1088216**
* **Linha Citros 1 Litro Limão**
* **PET**
* **1 Litro**
* **Limão**
* **7,00**



10) Verifique o conteúdo da tabela selecionando as 1000 primeiras linhas superiores:



**Alterando registros**

11) Crie uma nova consulta. Edite os comandos abaixo:

INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

('544931', 'Frescor do Verão - 350 ml - Limão', 'PET', '350 ml','Limão',3.20)

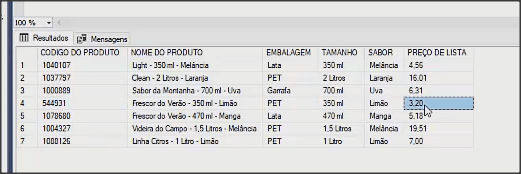
INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

('1078680', 'Frescor do Verão - 470 ml - Manga', 'Lata', '470 ml','Manga',5.18)

12) Execute e confira se estes novos registros foram incluídos na tabela:



13) Altere estes dois registros novos. Para isto, crie uma nova consulta e digite:

UPDATE [TABELA DE PRODUTOS] SET

[EMBALAGEM] = 'Lata',

[PREÇO DE LISTA] = 2.46

WHERE [CODIGO DO PRODUTO] = '544931'

UPDATE [TABELA DE PRODUTOS] SET

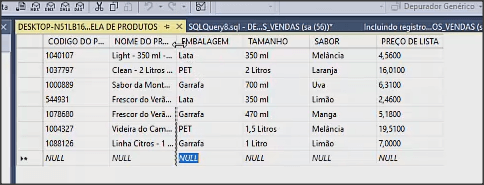
[EMBALAGEM] = 'Garrafa'

WHERE [CODIGO DO PRODUTO] = '1078680'

14) Execute e confira se estes novos registros foram alterados na tabela:



15) Você pode alterar o registros através da edição diretamente na tabela pelo *grid*. Para isso, clicando com o botão da direita do mouse sobre **Tabela de Produtos** e selecionando a opção **Editar 200 Linhas Superiores**, você pode modificar os registros:



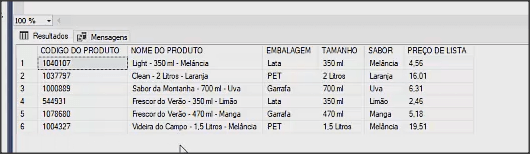
**Excluindo registros**

16) Crie uma nova consulta. Edite os comandos abaixo:

DELETE FROM [TABELA DE PRODUTOS]

WHERE [CODIGO DO PRODUTO] = '1088216'

17) Execute e verifique que o registro escolhido foi removido:



**Chave primária**

18) Crie uma nova consulta. Edite os comandos abaixo:

INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

('788975', 'Pedaços de Frutas - 1,5 Litros - Maça', 'PET', '1,5 Litros', 'Maça', 18.01)

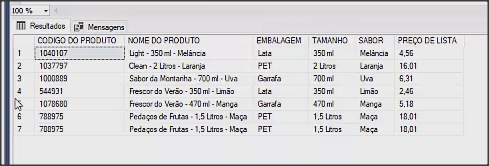
INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

VALUES

('788975', 'Pedaços de Frutas - 1,5 Litros - Maça', 'PET', '1,5 Litros', 'Maça', 18.01)

19) Ao executar, você irá iremos observar que dois registros foram gravados na tabela com códigos repetidos:



20) Crie a chave primária para evitar que isso acima aconteça. Mas antes, exclua estes registros repetidos, digitando e executando o comando:

DELETE FROM [TABELA DE PRODUTOS] WHERE [CODIGO DO PRODUTO] = '788975'

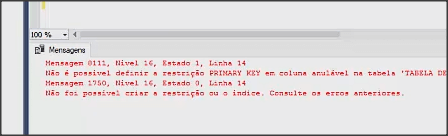
21) Digite o comando de criação da chave primária. Digite:

ALTER TABLE [TABELA DE PRODUTOS]

ADD CONSTRAINT PK\_PRODUTOS

PRIMARY KEY CLUSTERED ([CODIGO DO PRODUTO])

22) Execute o comando. Você terá uma mensagem de erro:



Isto acontece por causa da propriedade **NULL** dos campos que estão virando chaves primárias.

23) Digite o comando abaixo:

INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [PREÇO DE LISTA])

VALUES

('788975', 'Pedaços de Frutas - 1,5 Litros - Maça', 'PET', '1,5 Litros', 18.01)

Nele, você não irá incluir dados para a coluna **SABOR**.

24) Consulte agora a tabela. Veja que a coluna **SABOR** para este novo campo está preenchido com **NULL**:



**Resolvendo o problema da chave primária**

25) Crie uma nova consulta. Edite os comandos abaixo:

ALTER TABLE [TABELA DE PRODUTOS]

ALTER COLUMN {CODIGO DO PRODUTO]

VARCHAR(10) NOT NULL

E execute o passo 21 novamente.

26) Execute o comando de inclusão novamente:

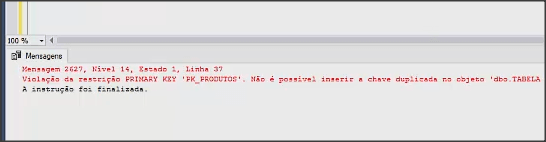
INSERT INTO [TABELA DE PRODUTOS]

([CODIGO DO PRODUTO], [NOME DO PRODUTO], [EMBALAGEM], [TAMANHO], [SABOR], [PREÇO DE LISTA])

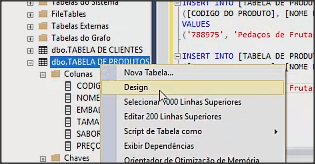
VALUES

('788975', 'Pedaços de Frutas - 1,5 Litros - Maça', 'PET', '1,5 Litros', 'Maça', 18.01)

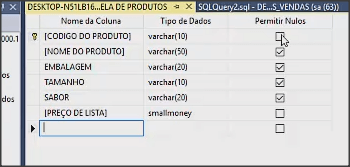
27) Execute-o novamente. Você terá o erro por causa da chave primária, não permitindo dois produtos com o mesmo código:



28) Clique com o botão da direita do mouse sobre a tabela **Tabela de Produtos** e escolha a opção **Designer**:



29) Você pode ver a representação da chave primária com a propriedade **Permitir Nulos** desmarcada e um símbolo de uma chave ao lado do nome do campo **[CODIGO DO PRODUTO]**:



**Manipulando datas e campos lógicos**

30) Crie uma nova consulta. Edite os comandos abaixo para criar a chave primária da tabela de clientes para o campo CPF:

ALTER TABLE [TABELA DE CLIENTES]

ALTER COLUMN [CPF]

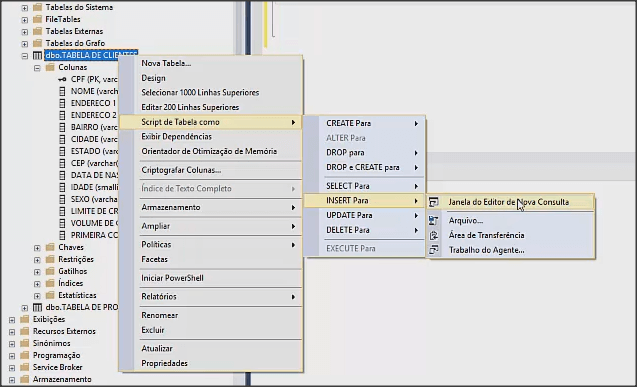
VARCHAR(11) NOT NULL

ALTER TABLE [TABELA DE CLIENTES]

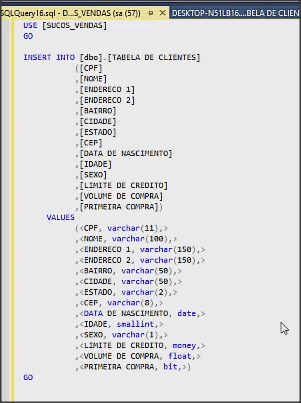
ADD CONSTRAINT PK\_CLIENTES

PRIMARY KEY CLUSTERED ([CPF])

31) Insira registros nesta tabela. Mas, para facilitar a digitação do comando de inserção, clique com o botão da direita do mouse sobre a **Tabela de Clientes** e escolha a opção **Script de Tabela como --> INSERT Para --> Janela do Editor de Nova Consulta**:



32) O comando de inclusão de registro é preenchido, bastando apenas que você complete com os dados:



33) Altere o comando digitando:

INSERT INTO [dbo].[TABELA DE CLIENTES]

([CPF]

,[NOME]

,[ENDERECO 1]

,[ENDERECO 2]

,[BAIRRO]

,[CIDADE]

,[ESTADO]

,[CEP]

,[DATA DE NASCIMENTO]

,[IDADE]

,[SEXO]

,[LIMITE DE CREDITO]

,[VOLUME DE COMPRA]

,[PRIMEIRA COMPRA])

VALUES

('00300000001'

,'João da Silva'

,'Rua Projetada A número 10'

,''

,'Centro'

,'Rio de Janeiro'

,'RJ'

,'20000000'

,'1990-10-25'

,27

,'M'

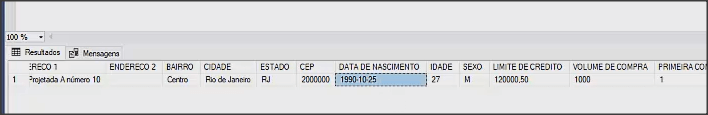
,120000.50

,1000

,1)

Veja neste comando a forma com que o campo do tipo **Data** e **Bit** foram incluídos.

34) Execute o comando e verifique o conteúdo da tabela:

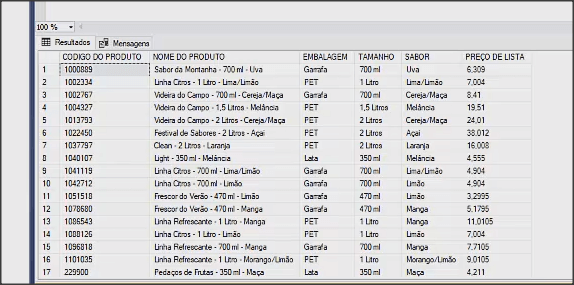


### Consultando os dados

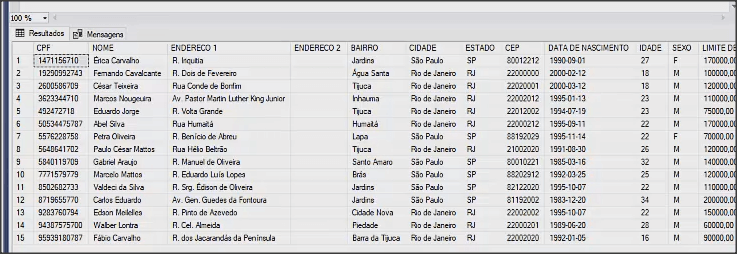
1) Acesse o **Management Studio**.

2) Abra o arquivo **Incluindo dados tabela produtos e clientes.sql** e execute o script. Diversos registros serão incluídos na tabela de produtos e de clientes.

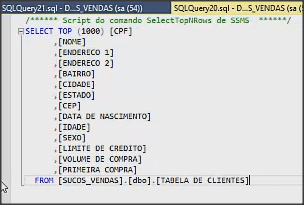
3) Se você verificar a tabela de produtos, terá:



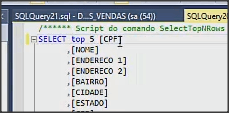
4) O mesmo para a tabela de clientes:

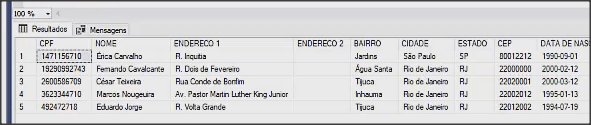


5) Quando listamos o conteúdo das tabelas, o SQL Server Management Studio escreveu justamente o comando para consulta de dados:



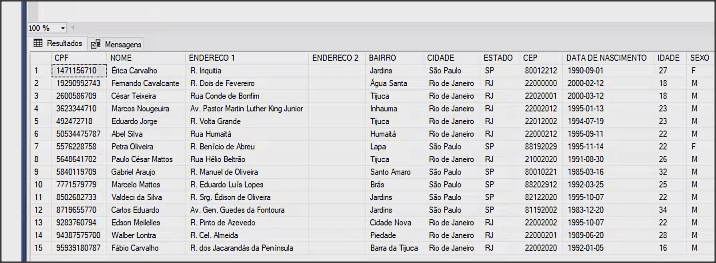
6) Se editarmos o comando acima e, no lugar de TOP 1000 você modificar para TOP 5, verá apenas 5 registros:





7) Se, no lugar dos campos, colocar um asterisco (como abaixo) verá a mesma coisa se estivesse digitado todos os campos.

SELECT \* FROM [TABELA DE CLIENTES]



8) Você pode restringir as colunas a serem exibidas. Digite o comando abaixo e execute-o.

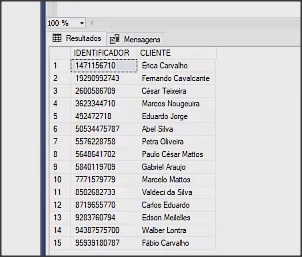
SELECT [CPF], [NOME] FROM [TABELA DE CLIENTES]



9) Você pode mudar o nome das colunas criando o que é chamado de **ALIAS**:

SELECT [CPF] AS IDENTIFICADOR, [NOME] AS CLIENTE

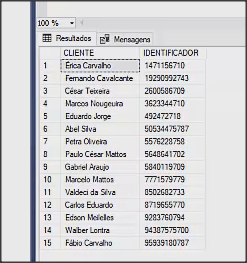
FROM [TABELA DE CLIENTES]



10) Ou também modificar a ordem de exibição das colunas no resultado da consulta:

SELECT [NOME] AS CLIENTE, [CPF] AS IDENTIFICADOR

FROM [TABELA DE CLIENTES]

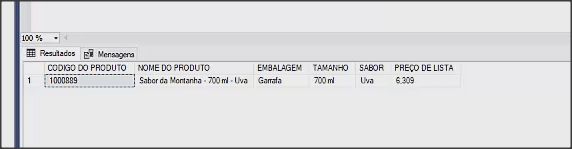


### Filtrando registros

11) Para limitar o resultado da consulta, inclua uma condição, muito parecida quando você aprendeu a excluir e alterar registros de uma tabela. Digite:

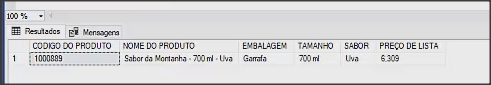
SELECT \* FROM [TABELA DE PRODUTOS] WHERE [CODIGO DO PRODUTO] = '1000889'

12) Você verá apenas o registro selecionado:



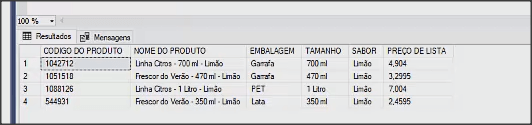
13) Você pode limitar a seleção usando como critério qualquer outro campo. Digite:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Uva'



14) Se você limitar para verificar somente sabores **Limão**, verá mais registros. Digite:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Limão'



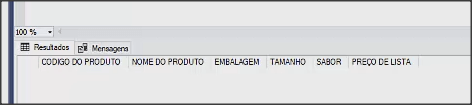
15) E qualquer condição de **WHERE** vale para os comandos **UPDATE** e **DELETE**. Digite:

DELETE FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Limão'

Você verá na mensagem de retorno que 4 registros foram afetados.

16) Se você executar novamente o comando de seleção, nada será retornado:

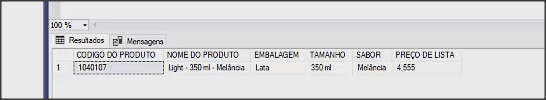
SELECT \* FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Limão'



### Filtrando usando maior, menor, diferente

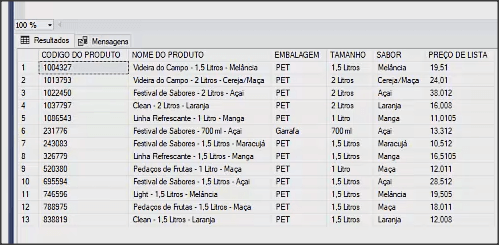
17) Você pode fazer filtros envolvendo números. Digite:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] = 4.555



18) Você pode verificar os produtos que custam mais que 10. Digite:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] > 10



19) Ou menores que 10:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] < 10



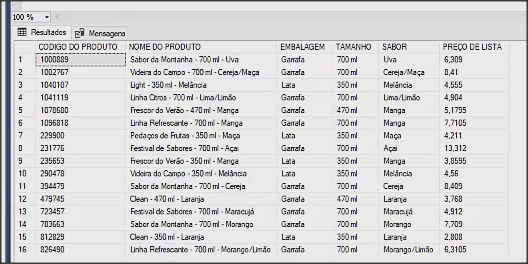
20) Você pode usar o maior e igual, ou menor ou igual para incluir o critério de seleção no resultado:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] >= 4.555



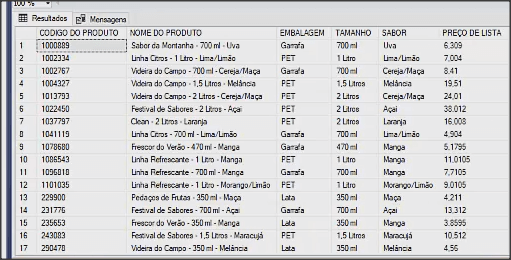
21) Você pode usar o maior e menor para condições que envolvam strings. O critério de determinar quem é maior ou menor será o de ordem alfabética:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE EMBALAGEM <= 'Lata'



22) Há também o critério de selecionar alguém diferente de um valor. Para isso, use o símbolo **<>**:

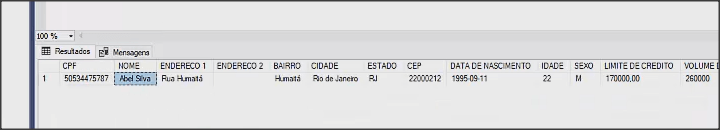
SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] <> 4.555



### Filtrando por datas

23) Você pode usar as datas como critério de seleção. O SQL Server irá usar a ordem definida no calendário para determinar quem é maior ou menor. Digite:

SELECT \* FROM [TABELA DE CLIENTES] WHERE [DATA DE NASCIMENTO] = '1995-09-11'



24) Ou então:

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



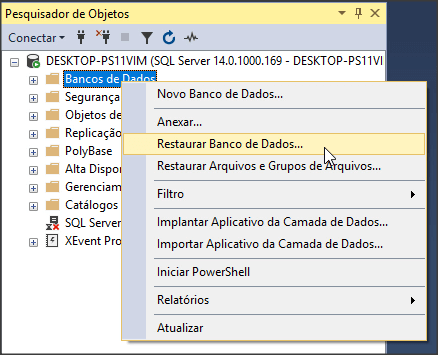
25) Você pode usar algumas funções aplicadas à data que determinam condições sobre o dia, mês ou ano. Veja o exemplo abaixo:

SELECT \* FROM [TABELA DE CLIENTES] WHERE YEAR([DATA DE NASCIMENTO]) < 1995

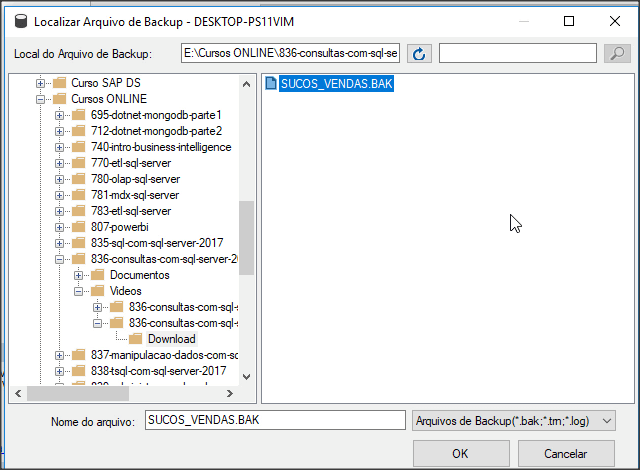


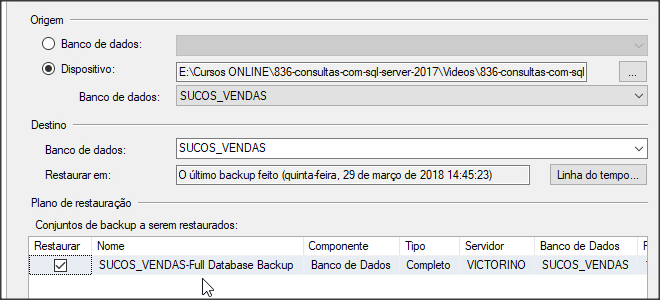
Nesta atividade, você verá como preparar o seu ambiente, para poder realizar as tarefas e prosseguir com o treinamento. Para isso, baixe [aqui](https://s3.amazonaws.com/caelum-online-public/836-consultas-com-sql-server-2017/01/Downloads+-+Aula+1.zip) os arquivos necessários e siga os passos abaixo.

1) Vá no **SQL Server Management Studio 2017** e conecte-se ao seu servidor. Após isso, na árvore à esquerda, clique com o botão da direita do mouse sobre **Banco de Dados** e selecione a opção **Restaurar Banco de Dados**:

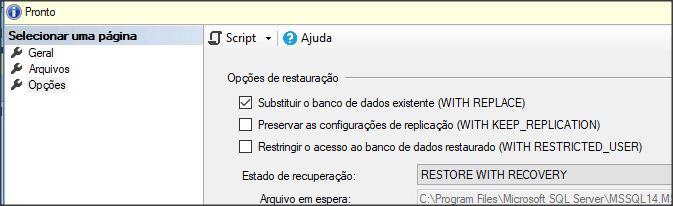


2) Selecione o arquivo a ser usado para recuperação do backup:

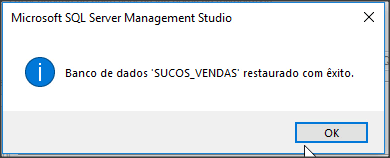




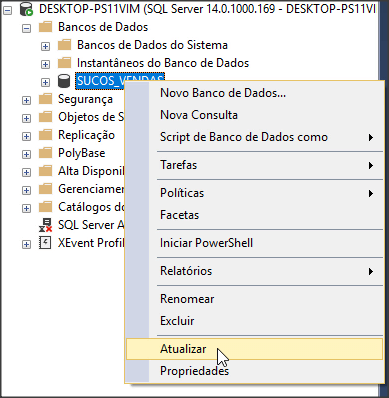
3) Selecione a opção **Substituir banco de dados existente**:



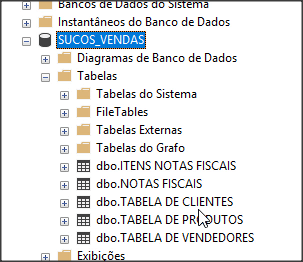
4) Clique em **OK**. O banco de dados será restaurado:



5) Para você se certificar que a restauração funcionou, clique em **Atualizar**, após clicar com o botão da direita do mouse sobre o banco de dados:

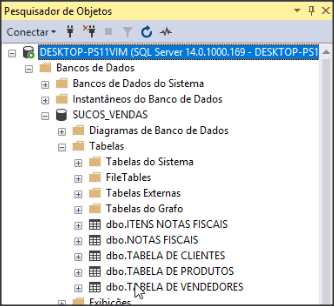


6) E confira se as tabelas do banco de dados são exibidas:



**Conhecendo o banco de dados**

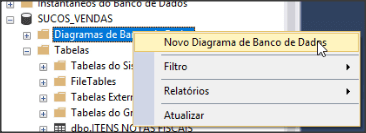
1) Acesse o **SQL Server Management Studio 2017** e verifique as tabelas do banco de dados de vendas da empresa de suco de frutas:

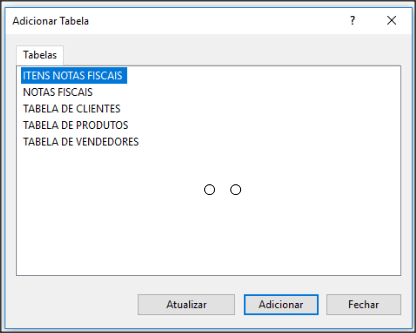


2) Há as tabelas:

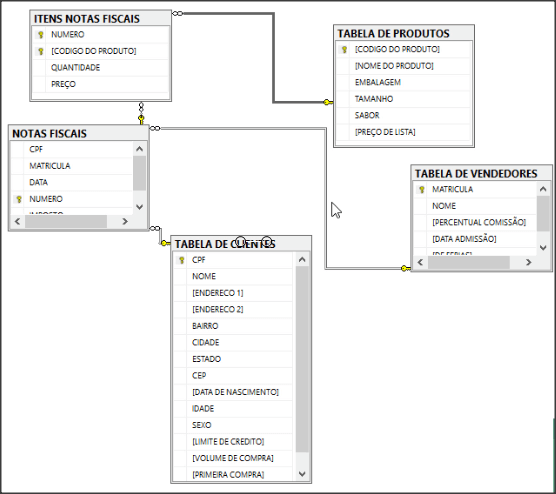
* **TABELA DE VENDEDORES** - Lista de vendedores que efetuaram as vendas.
* **TABELA DE PRODUTOS** - Lista dos produtos oferecidos pela empresa.
* **TABELA DE CLIENTES** - Lista de clientes que compram os produtos da empresa.
* **NOTAS FISCAIS** - Notas fiscais das vendas feitas pela empresa.
* **ITEMS NOTAS FISCAIS** - Itens das notas fiscais.

3) Uma forma rápida de visualizar as tabelas e seus relacionamentos é indo em **Diagrama de Banco de Dados** e, clicando com o botão da direita do mouse sobre esta opção, escolha a opção **Novo Diagrama de Banco de Dados**. Você terá a lista de todas as tabelas:





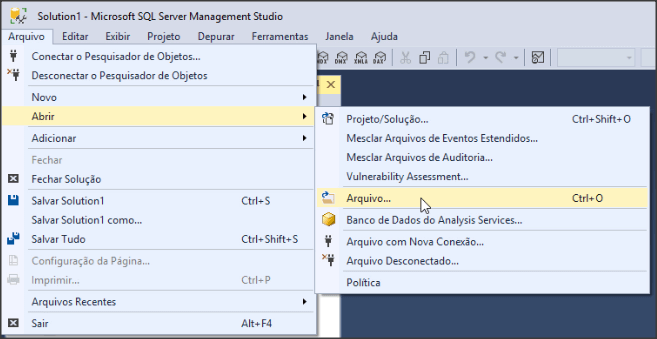
4) Selecione todas as tabelas e clique em **Adicionar**. Você terá a representação das tabelas graficamente:



5) Observe os campos de cada tabela com os seus respectivos tipos e as relações entre as tabelas.

**Revisando consultas**

6) Vá no **SQL Server Management Studio 2017** e selecione no menu: **Arquivo --> Abrir --> Arquivo...**:



7) Selecione o arquivo **Revisao Consultas.sql**:



8) Reveja as consultas já vistas no curso de **Introdução ao SQL Server**. Digite:

SELECT [CPF]

,[NOME]

,[ENDERECO 1]

,[ENDERECO 2]

,[BAIRRO]

,[CIDADE]

,[ESTADO]

,[CEP]

,[DATA DE NASCIMENTO]

,[IDADE]

,[SEXO]

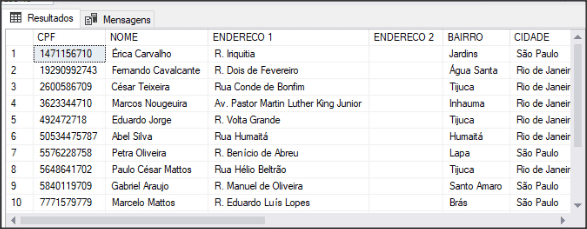
,[LIMITE DE CREDITO]

,[VOLUME DE COMPRA]

,[PRIMEIRA COMPRA]

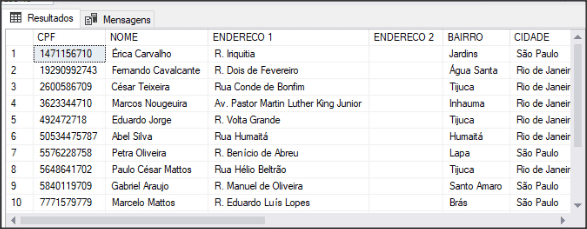
FROM [SUCOS\_VENDAS].[dbo].[TABELA DE CLIENTES]

Você terá a lista de clientes da **TABELA DE CLIENTES**:



9) Você pode substituir a lista dos campos por um asterisco:

SELECT \* FROM [TABELA DE CLIENTES]



10) Você pode escolher os campos a serem visualizados:

SELECT [CPF]

,[NOME]

FROM [TABELA DE CLIENTES]

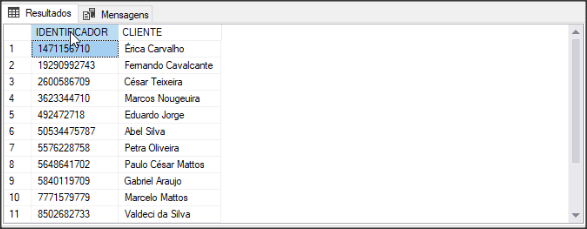


11) Você também pode criar um ***alias*** para cada coluna:

SELECT [CPF] AS IDENTIFICADOR

,[NOME] AS CLIENTE

FROM [TABELA DE CLIENTES]

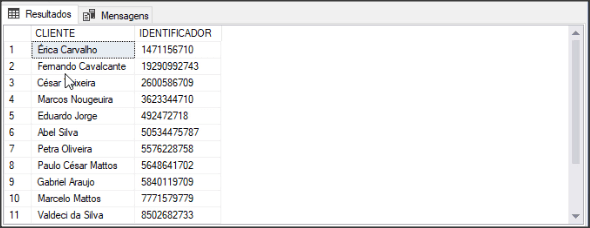


12) E pode trocar a ordem de exibição do resultado:

SELECT [NOME] AS CLIENTE

,[CPF] AS IDENTIFICADOR

FROM [TABELA DE CLIENTES]



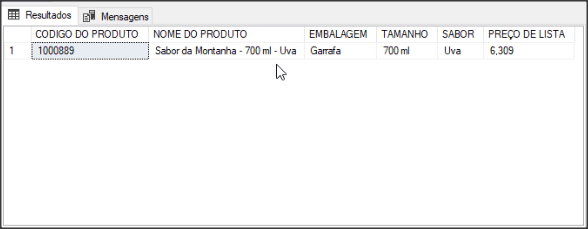
13) Você pode filtrar os dados exibidos pela consulta, usando o campo primário:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [CODIGO DO PRODUTO] = '1000889'



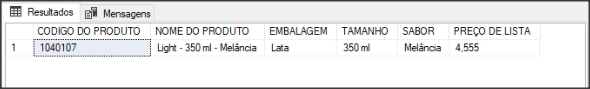
14) Mas também pode filtrar por qualquer outro campo:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Uva'



15) No caso de filtros usando valores, você não usa aspas simples no critério de busca:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] = 4.555



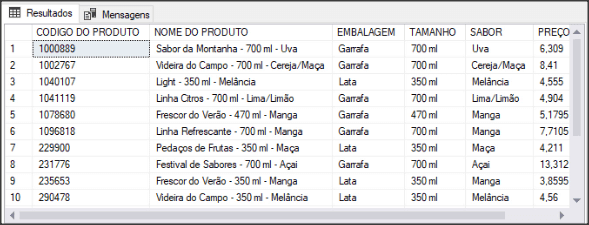
16) Use os símbolos de **>**, **>=**, **<** ou **<=** para filtrar os dados:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [PREÇO DE LISTA] > 10



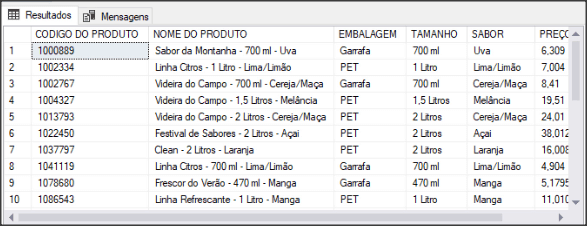
17) O critério de **>**, **>=**, **<** ou **<=** pode ser aplicado a um campo string. O SQL Server irá usar o critério de ordenação alfabética:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE EMBALAGEM <> 'Lata'



18) Você também pode aplicar filtro de datas. Para evitar problemas de critérios de representação das datas, use, no filtro, o formato **ANO-MÊS-DIA**:

SELECT \* FROM [TABELA DE CLIENTES] WHERE [DATA DE NASCIMENTO] = '1995-09-11'



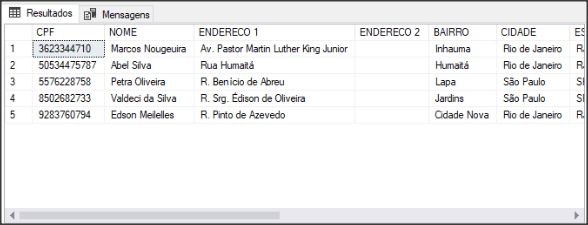
19) Você pode usar **>**, **>=**, **<** ou **<=** para as datas:

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



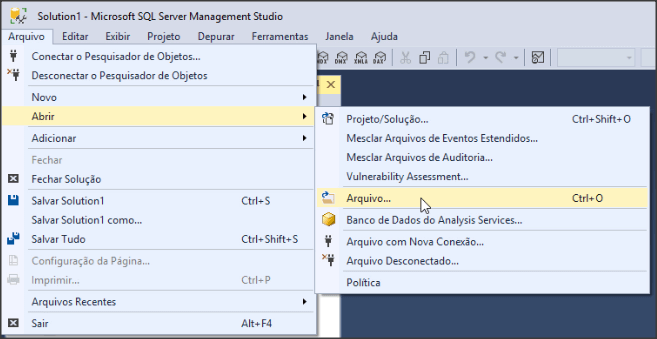
20) Nos critérios de filtro, você pode usar funções do SQL Server. Por exemplo, use a função **YEAR** aplicada ao campo **DATA DE NASCIMENTO**:

SELECT \* FROM [TABELA DE CLIENTES] WHERE YEAR([DATA DE NASCIMENTO]) = 1995

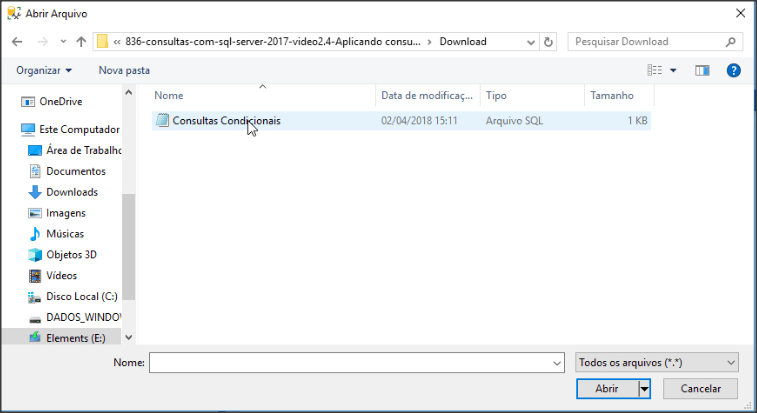


**Aplicando consultas condicionais no SQL Server**

21) Vá no **SQL Server Management Studio 2017** e selecione no menu: **Arquivo --> Abrir --> Arquivo...**:



22) Selecione o arquivo **Consultas Condicionais.sql**:



23) Use o **OR** para selecionar produtos que sejam de um determinado sabor ou de um determinado tamanho:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

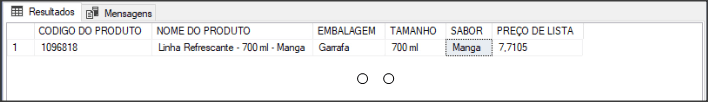
[SABOR] = 'Manga' OR [TAMANHO] = '700 ml'



24) Já o comando **AND** listará os produtos que sejam de um determinado sabor e também de um determinado tamanho:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

[SABOR] = 'Manga' AND [TAMANHO] = '700 ml'



25) O comando **NOT**, na frente da condição, retorna o contrário do que a condição busca:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

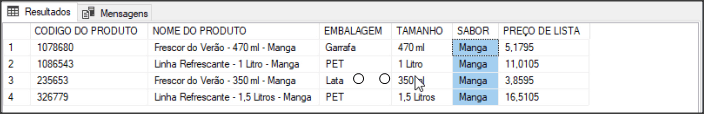
NOT ([SABOR] = 'Manga' AND [TAMANHO] = '700 ml')



26) O **NOT** pode ser aplicado somente a uma parte da condição de filtro:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

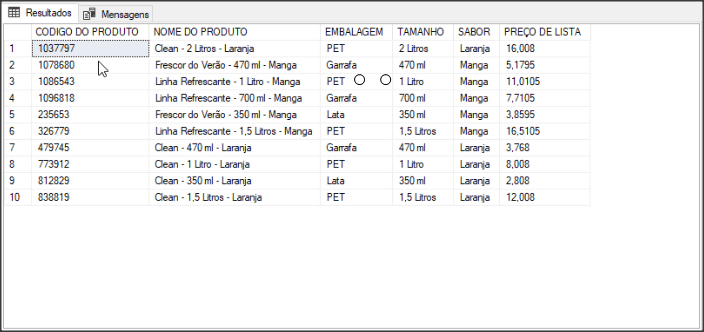
[SABOR] = 'Manga' AND NOT ( [TAMANHO] = '700 ml')



27) O comando **IN** busca o registro através de um conjunto de valores:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

[SABOR] IN ('Manga', 'Laranja')



Esse comando seria equivalente a:

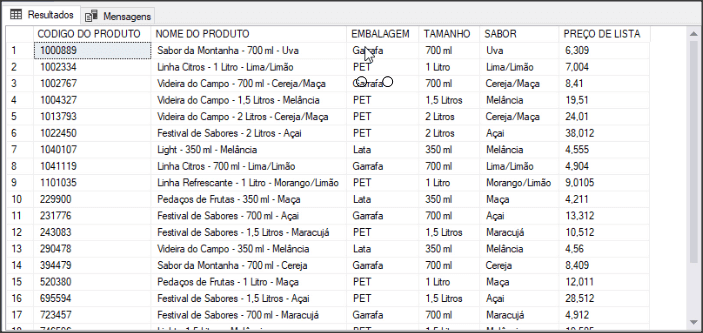
SELECT \* FROM [TABELA DE PRODUTOS] WHERE

[SABOR] = 'Manga' OR [SABOR] ='Laranja'

28) Você pode aplicar o **NOT** sobre a condição com **IN**:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

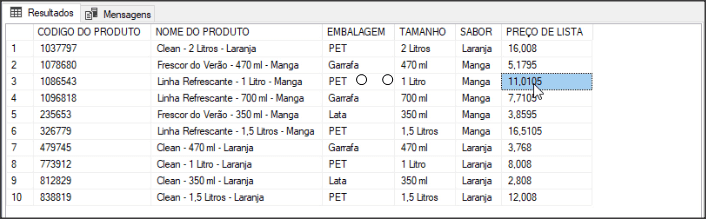
[SABOR] NOT IN ('Manga', 'Laranja')



29) É possível colocar junto com o **IN** condições usando **>**, **>=**, **<** ou **<=**:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

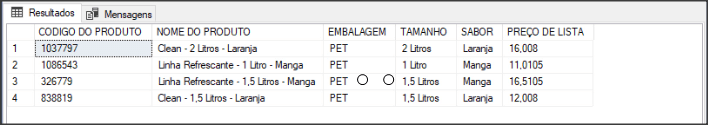
[SABOR] IN ('Manga', 'Laranja') AND [PREÇO DE LISTA] > 10



30) Há o comando **BETWEEN**, onde você pode selecionar entre dois valores:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

[SABOR] IN ('Manga', 'Laranja') AND [PREÇO DE LISTA] BETWEEN 10 AND 13



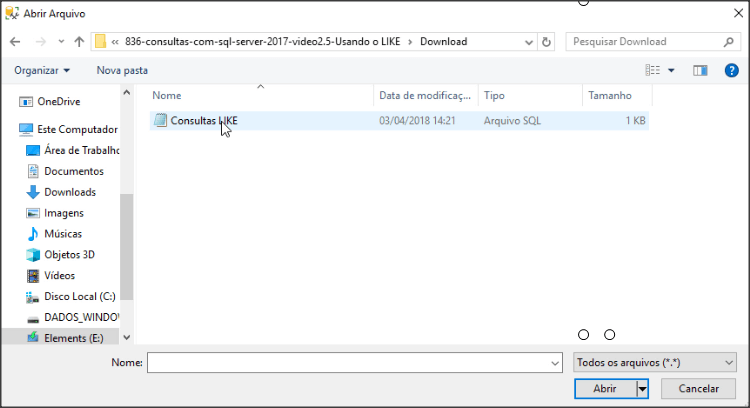
Que seria equivalente ao comando:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE

[SABOR] IN ('Manga', 'Laranja') AND [PREÇO DE LISTA] >= 10 AND [PREÇO DE LISTA] <= 13

**Usando o LIKE**

31) Vá no **SQL Server Management Studio 2017** e selecione no menu: **Arquivo --> Abrir --> Arquivo...**. Em seguida, selecione o arquivo **Consultas LIKE.sql**:



32) Execute o comando abaixo:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [NOME DO PRODUTO] LIKE '%Litros%'

Aqui, você irá achar todos os registros cujo campo **[NOME DO PRODUTO]** possua dentro do seu descritor a palavra **Litros**, seja no início, no fim ou no meio do descritor:



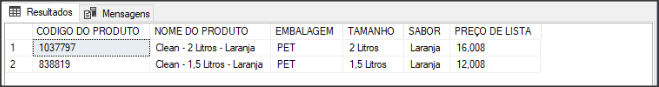
33) Você pode buscar apenas os campos usando o critério de apenas o campo começar com a palavra **Litros**:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [NOME DO PRODUTO] LIKE 'Litros%'

34) Também podemos mesclar a condição **LIKE** com outras condições:

SELECT \* FROM [TABELA DE PRODUTOS] WHERE [NOME DO PRODUTO] LIKE '%Litros%'

AND [SABOR] = 'Laranja'

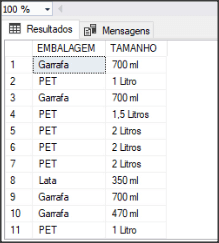


### Usando DISTINCT para visualizar a tabela

1) Abra o arquivo **Consultas DISTINCT 1.sql** no **SQL Server Management Studio 2017**.

2) Execute a consulta:

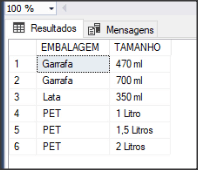
SELECT EMBALAGEM, TAMANHO FROM [TABELA DE PRODUTOS]



Note que há diversos registros com **EMBALAGEM** e **TAMANHO** repetidos.

3) Agora, execute o mesmo comando com a cláusula **DISTINCT**:

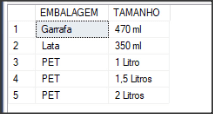
SELECT DISTINCT EMBALAGEM, TAMANHO FROM [TABELA DE PRODUTOS]



4) Os filtros aplicados até agora podem ser executados em conjunto com os comandos SQL com a cláusula **DISTINCT**, mesmo que o campo usado para a condição não esteja sendo exibido no resultado da consulta:

SELECT DISTINCT EMBALAGEM, TAMANHO FROM [TABELA DE PRODUTOS] WHERE

[SABOR] = 'Laranja'

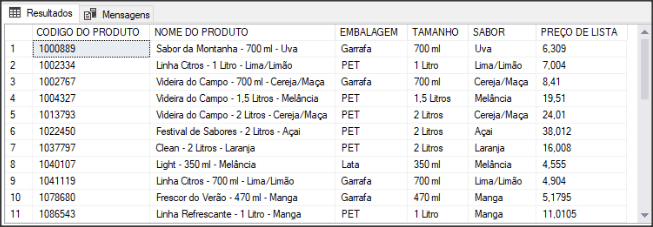


### Limitando a saída da consulta

5) Abra o arquivo **Consultas DISTINCT 2.sql**.

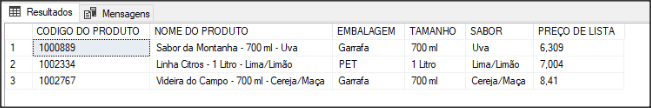
6) Execute a seleção de todos os produtos:

SELECT \* FROM [TABELA DE PRODUTOS]



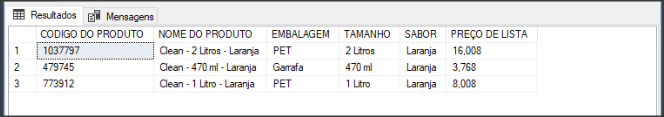
7) Limite e olhe os primeiros 3 produtos desta lista:

SELECT TOP 3 \* FROM [TABELA DE PRODUTOS]



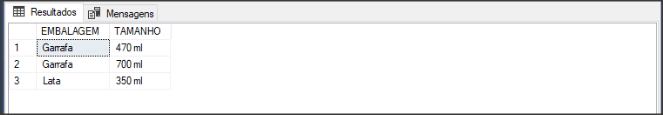
8) Limite a consulta, utilizando filtros, como mostrado abaixo:

SELECT TOP 3 \* FROM [TABELA DE PRODUTOS] WHERE [SABOR] = 'Laranja'



9) Também é possível mesclar o **TOP** com o **DISTINCT** como mostrado abaixo:

SELECT DISTINCT TOP 3 EMBALAGEM, TAMANHO FROM [TABELA DE PRODUTOS]

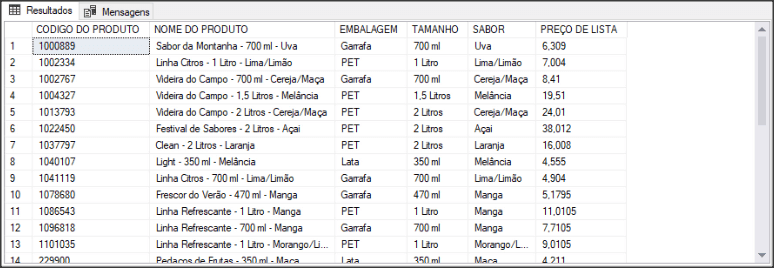


### Ordenando a saída da consulta

10) Abra o arquivo **Ordenando a saída da consulta.sql**.

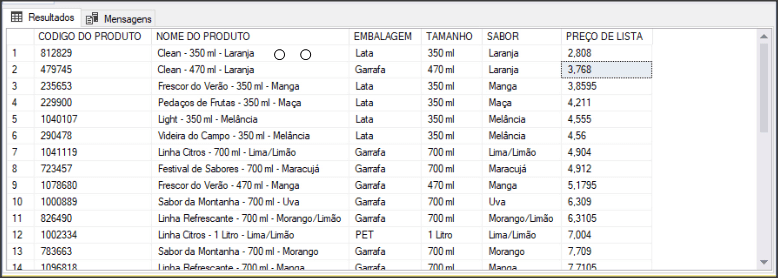
11) Execute o comando abaixo para listar todos os produtos:

SELECT \* FROM [dbo].[TABELA DE PRODUTOS]



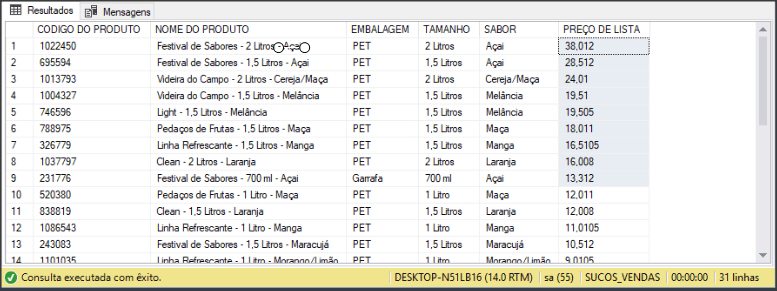
12) Visualize esta mesma lista, mas ordenada pelo preço:

SELECT \* FROM [TABELA DE PRODUTOS] ORDER BY [PREÇO DE LISTA]



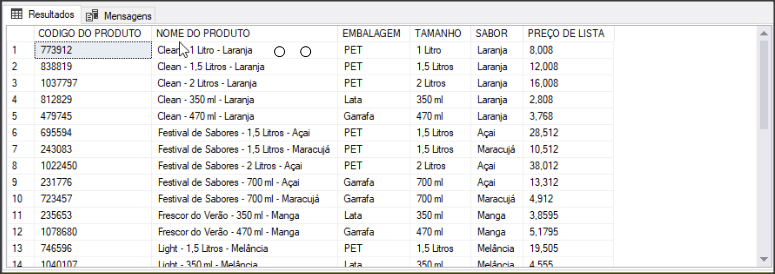
13) Na consulta acima, você vê os dados ordenados do menor para o maior. Se quiser ao contrário, use a cláusula **DESC**:

SELECT \* FROM [TABELA DE PRODUTOS] ORDER BY [PREÇO DE LISTA] DESC



14) Você pode usar como critério de ordenação um campo string. Neste caso, o SQL Server usará como critério de ordenação a ordem alfabética:

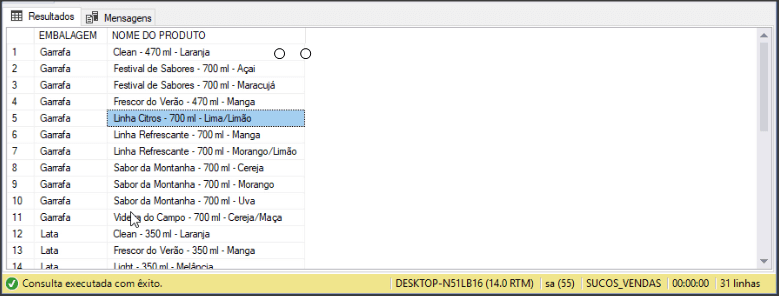
SELECT \* FROM [TABELA DE PRODUTOS] ORDER BY [NOME DO PRODUTO]



15) Bem como ordenar usando critérios em conjunto:

SELECT [EMBALAGEM], [NOME DO PRODUTO] FROM [TABELA DE PRODUTOS]

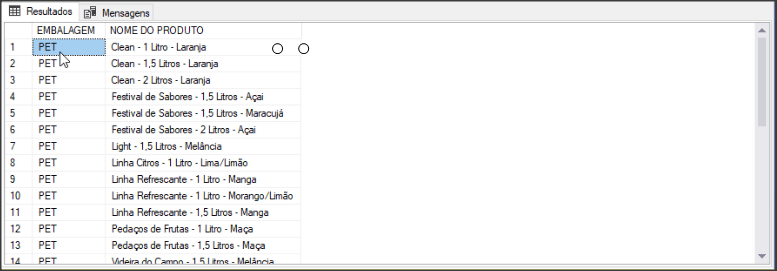
ORDER BY [EMBALAGEM], [NOME DO PRODUTO]



16) Inclusive a direção da ordenação pode ser diferente. Uma para cada tipo de campo:

SELECT [EMBALAGEM], [NOME DO PRODUTO] FROM [TABELA DE PRODUTOS]

ORDER BY [EMBALAGEM] DESC, [NOME DO PRODUTO] ASC



### Agrupando os resultados

17) Abra o arquivo **Agrupando os resultados.sql**.

18) Liste todos os clientes:

SELECT \* FROM [dbo].[TABELA DE CLIENTES]



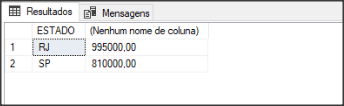
19) Agora, selecione somente o estado e o limite de crédito:

SELECT ESTADO, [LIMITE DE CREDITO] FROM [dbo].[TABELA DE CLIENTES]



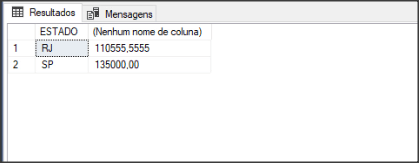
20) Calcule o total de limite de crédito por estado. Neste caso, use o comando **GROUP BY** para obter este resultado:

SELECT ESTADO, SUM([LIMITE DE CREDITO]) FROM [TABELA DE CLIENTES] GROUP BY ESTADO



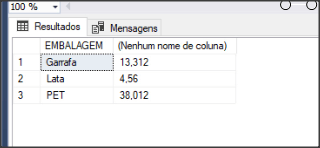
21) Você pode obter também a média de crédito em cada estado:

SELECT ESTADO, AVG([LIMITE DE CREDITO]) FROM [TABELA DE CLIENTES] GROUP BY ESTADO



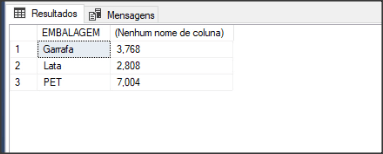
22) Do preço de lista dos produtos, obtenha os mais caros por embalagem:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM



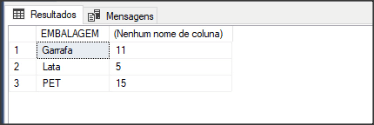
23) Ou então o menor:

SELECT EMBALAGEM, MIN([PREÇO DE LISTA]) FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM



24) O comando **COUNT(\*)** conta o número de ocorrências. Por exemplo, para saber quantos produtos há por cada tipo de embalagem, execute:

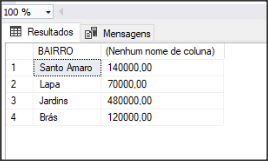
SELECT EMBALAGEM, COUNT(\*) FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM



25) E você pode juntar os comandos de **GROUP BY** com os filtros já apresentados anteriormente:

SELECT BAIRRO, ESTADO, SUM([LIMITE DE CREDITO]) FROM [TABELA DE CLIENTES]

WHERE [ESTADO] = 'SP' GROUP BY BAIRRO, ESTADO



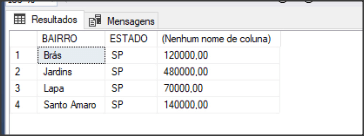
### Usando a condição HAVING

26) Abra o arquivo **Usando Having.sql**.

27) Execute o seguinte comando:

SELECT BAIRRO, ESTADO, SUM([LIMITE DE CREDITO]) FROM [TABELA DE CLIENTES]

WHERE [ESTADO] = 'SP' GROUP BY BAIRRO, ESTADO

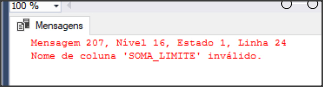


28) Agora, limite esta consulta aos valores maiores que 140000. O natural seria aplicar um filtro sobre o **SUM([LIMITE DE CREDITO])**:

SELECT BAIRRO, ESTADO, SUM([LIMITE DE CREDITO]) AS SOMA\_LIMITE FROM [TABELA DE CLIENTES]

WHERE [ESTADO] = 'SP' AND SOMA\_LIMITE >= 140000 GROUP BY BAIRRO, ESTADO

Nesta consulta, foi aplicado um alias para **SUM([LIMITE DE CREDITO])**, que foi utilizado no filtro **SOMA\_LIMITE >= 140000**. Mas, ao rodar esta consulta, você terá:

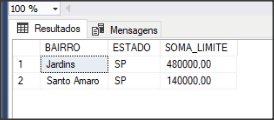


29) Para isso, use o **HAVING**. Digite o seguinte comando:

SELECT BAIRRO, ESTADO, SUM([LIMITE DE CREDITO]) AS SOMA\_LIMITE FROM [TABELA DE CLIENTES]

WHERE [ESTADO] = 'SP' GROUP BY BAIRRO, ESTADO HAVING SUM([LIMITE DE CREDITO]) >= 140000

Você terá:



30) Olhe outra consulta, agora associada à lista de clientes:

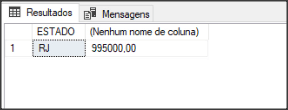
SELECT \* FROM [dbo].[TABELA DE CLIENTES]



31) Observe o limite de crédito para cada estado, limitados a valores maiores ou iguais a 850000:

SELECT ESTADO, SUM([LIMITE DE CREDITO]) FROM [TABELA DE CLIENTES] GROUP BY ESTADO

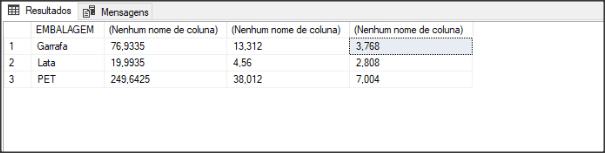
HAVING SUM([LIMITE DE CREDITO]) >= 850000



32) Junte condições no **HAVING**. Veja a consulta abaixo que ao mesmo tempo obtém a soma (**SUM**), o **MAX** e o **MIN** de preços:

SELECT EMBALAGEM, SUM([PREÇO DE LISTA]), MAX([PREÇO DE LISTA]), MIN([PREÇO DE LISTA])

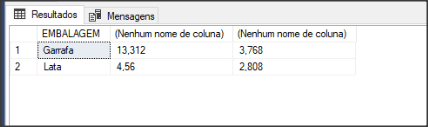
FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM



33) E então aplique o **HAVING** sobre esta consulta:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]), MIN([PREÇO DE LISTA]) FROM [TABELA DE PRODUTOS]

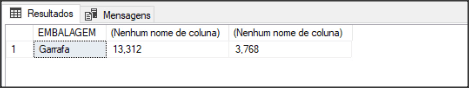
GROUP BY EMBALAGEM HAVING SUM([PREÇO DE LISTA]) <= 80



34) Você pode, inclusive, fazer condições compostas:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]), MIN([PREÇO DE LISTA]) FROM [TABELA DE PRODUTOS]

GROUP BY EMBALAGEM HAVING SUM([PREÇO DE LISTA]) <= 80 AND MAX([PREÇO DE LISTA]) >= 6

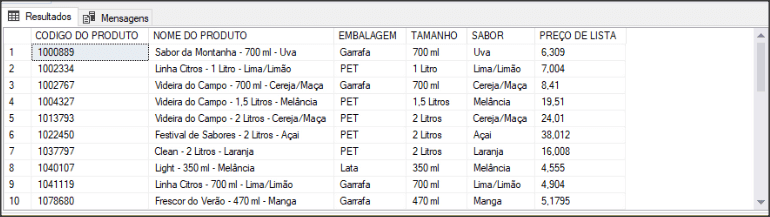


### Usando condições para classificar resultados

35) Abra o arquivo **Classificando Resultados.sql**.

36) Verifique a lista de produtos:

SELECT \* FROM [TABELA DE PRODUTOS]



37) Classifique os produtos em **CAROS** (preço maior que 12), **EM CONTA** (preço entre 7 e 12) ou **BARATOS** (preço menor que 7):

SELECT [NOME DO PRODUTO],

CASE WHEN [PREÇO DE LISTA] >= 12 THEN 'PRODUTO CARO'

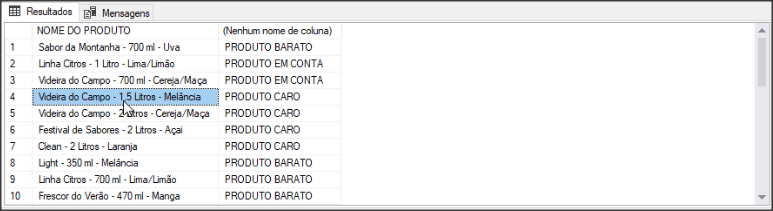
WHEN [PREÇO DE LISTA] >= 7 AND [PREÇO DE LISTA] < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO' END

FROM

[TABELA DE PRODUTOS]

Você terá:



38) Você pode juntar uma função de **GROUP BY** com a classificação. Como ela faz parte da agregação, a regra de construção da classificação deve estar presente na propriedades do **GROUP BY**. Veja abaixo:

SELECT [NOME DO PRODUTO],

CASE WHEN [PREÇO DE LISTA] >= 12 THEN 'PRODUTO CARO'

WHEN [PREÇO DE LISTA] >= 7 AND [PREÇO DE LISTA] < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO' END,

AVG([PREÇO DE LISTA])

FROM

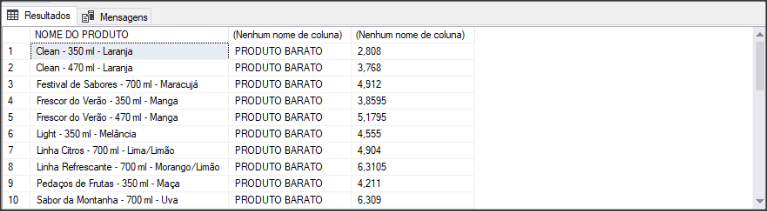
[TABELA DE PRODUTOS]

GROUP BY [NOME DO PRODUTO],

CASE WHEN [PREÇO DE LISTA] >= 12 THEN 'PRODUTO CARO'

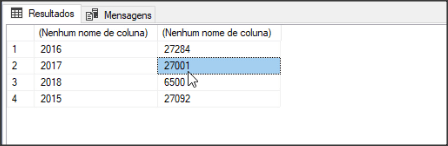
WHEN [PREÇO DE LISTA] >= 7 AND [PREÇO DE LISTA] < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO' END



39) Isso mostra que sempre que você usar uma fórmula ou função dentro da especificação do **SELECT**, e conter um agrupamento, esta função ou fórmula deve fazer parte do **GROUP BY**:

SELECT YEAR(DATA), COUNT(\*) FROM [NOTAS FISCAIS] GROUP BY YEAR(DATA)



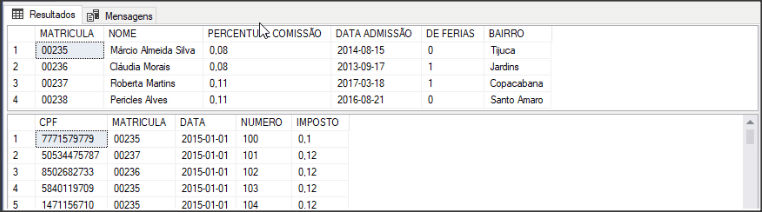
### Usando o INNER JOIN

1) Abra o arquivo **Usando INNER JOIN.sql**.

2) Olhe as tabelas de vendedores e notas fiscais:

SELECT \* FROM [TABELA DE VENDEDORES]

SELECT \* FROM [NOTAS FISCAIS]

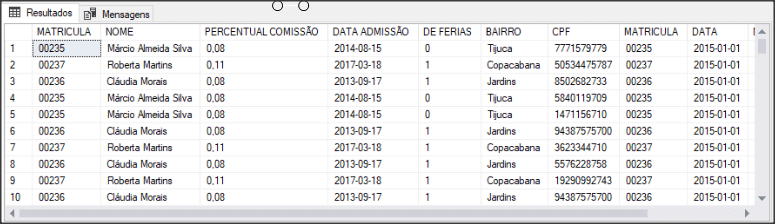


Há um campo em comum entre as duas tabelas, que é a **MATRICULA**. Por este campo é que você pode juntar as duas tabelas.

3) Execute:

SELECT \* FROM [TABELA DE VENDEDORES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA



Neste resultado há todos os campos das duas tabelas.

4) Quando você faz a igualdade:

[TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA

Tem que colocar o nome da tabela na frente do nome do campo. Isso porque se você usar:

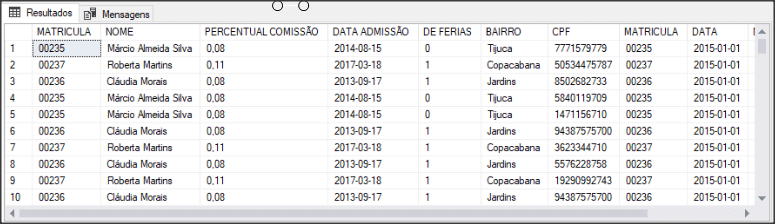
MATRICULA = MATRICULA

O SQL Server não saberá de que tabela pertence os campos **MATRICULA**.

5) Crie um alias, que pode ser um nome ou uma letra, para colocar ao lado do nome da tabela. O comando ficará assim:

SELECT \* FROM [TABELA DE VENDEDORES] A INNER JOIN

[NOTAS FISCAIS] B ON A.MATRICULA = B.MATRICULA



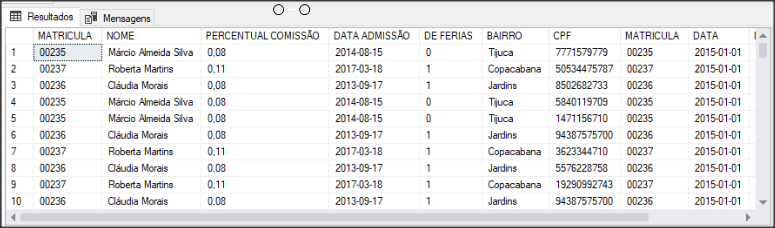
6) Selecione os campos a serem exibidos no **JOIN**. Você pode aplicar tudo o que já foi visto até agora nesta seleção (filtros, TOP, DISTINCT, GROUP BY, ORDER BY, etc). Por exemplo, conte o número de notas fiscais por cada vendedor:

SELECT [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME], COUNT(\*)

FROM [TABELA DE VENDEDORES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA

GROUP BY [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME]



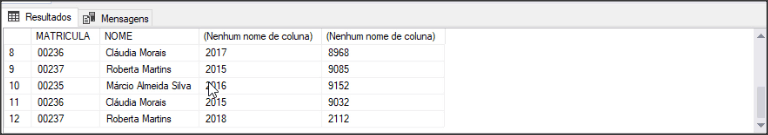
7) Melhore a consulta e veja o número de notas fiscais, mas dentro de cada ano:

SELECT [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME], YEAR(DATA),

COUNT(\*) FROM [TABELA DE VENDEDORES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA

GROUP BY [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME], YEAR(DATA)



8) Nesta consulta final, ajuste a consulta acima, mostrando por ordem de data e vendedor:

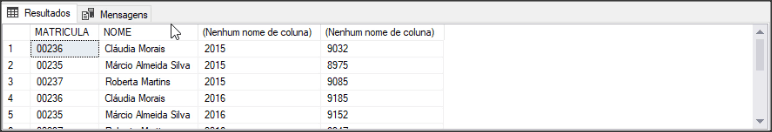
SELECT [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME], YEAR(DATA),

COUNT(\*) FROM [TABELA DE VENDEDORES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA

GROUP BY [TABELA DE VENDEDORES].MATRICULA, [TABELA DE VENDEDORES].[NOME], YEAR(DATA)

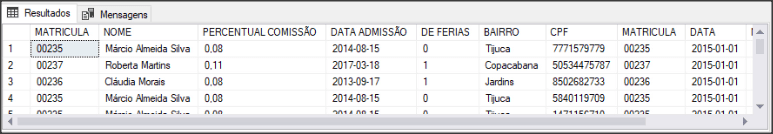
ORDER BY YEAR(DATA), [TABELA DE VENDEDORES].[NOME]



9) Existe uma forma antiga de fazer **JOIN**, que é considerá-lo como se fosse um filtro e declarar as tabelas que vão ser unidas após o **FROM**, com as mesmas separadas por vírgula:

SELECT \* FROM [TABELA DE VENDEDORES], [NOTAS FISCAIS]

WHERE [TABELA DE VENDEDORES].MATRICULA = [NOTAS FISCAIS].MATRICULA



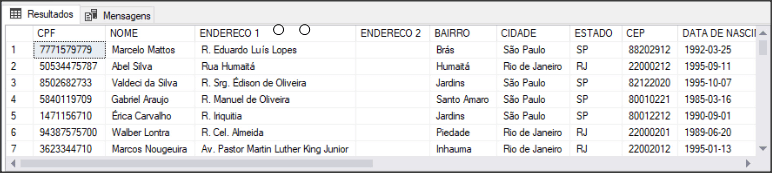
### Usando LEFT e RIGHT JOIN

10) Abra o arquivo **Usando LEFT e RIGHT JOIN.sql**.

11) Junte a tabelas de clientes com a tabelas de notas fiscais pelo campo **CPF**, que é o campo em comum das duas:

SELECT \* FROM [TABELA DE CLIENTES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE CLIENTES].CPF = [NOTAS FISCAIS].CPF

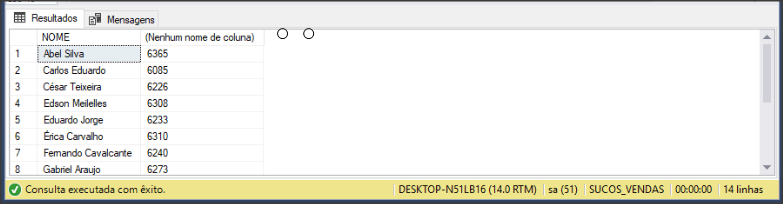


12) Aplicando o **COUNT(\*)** com **GROUP BY**, conte o número de notas fiscais por cada cliente:

SELECT [TABELA DE CLIENTES].[NOME], COUNT(\*) FROM [TABELA DE CLIENTES] INNER JOIN

[NOTAS FISCAIS] ON [TABELA DE CLIENTES].CPF = [NOTAS FISCAIS].CPF

GROUP BY [TABELA DE CLIENTES].[NOME]

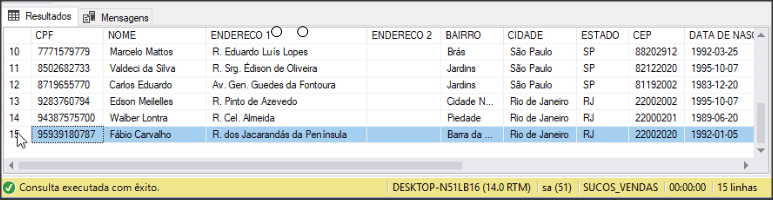


Note que esta consulta retornou 14 linhas:



13) Verifique quantos clientes há na tabela de clientes:

SELECT \* FROM [TABELA DE CLIENTES]



Esta consulta retorna 15 linhas:



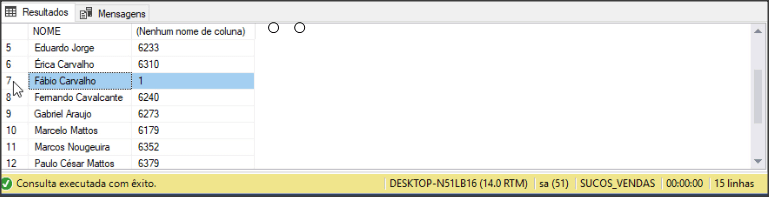
Isto significa que, quando você vê a tabela de clientes, há 15 linhas, mas quando agrupa a consulta, fazendo **JOIN** com a tabela de notas fiscais, há 14 clientes. Logo, existe um cliente que faz parte do cadastro e que nunca fez uma compra. Pelo **LEFT JOIN**, você pode podemos descobrir este cliente.

14) Para descobrir o cliente que nunca comprou, faça:

SELECT [TABELA DE CLIENTES].[NOME], COUNT(\*) FROM [TABELA DE CLIENTES] LEFT JOIN

[NOTAS FISCAIS] ON [TABELA DE CLIENTES].CPF = [NOTAS FISCAIS].CPF

GROUP BY [TABELA DE CLIENTES].[NOME]



O cliente que tem valor de nota fiscal igual a **1** é o cliente que nunca comprou. O valor **1** aparece porque o **COUNT(\*)** faz uma contagem de registros e este cliente aparece na consulta, apesar de não ter nenhuma nota. Por causa do **LEFT JOIN**, ele aparece apenas uma vez e por isso o **COUNT(\*)** escreve para ele o valor **1**.

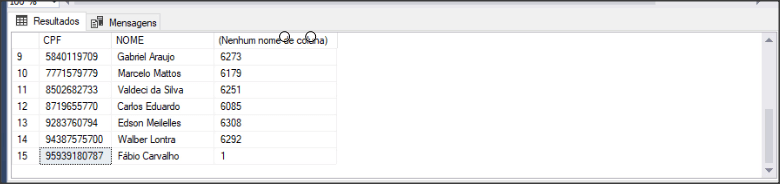
15) Para ter certeza deste resultado, coloque o CPF na consulta:

SELECT [TABELA DE CLIENTES].[CPF], [TABELA DE CLIENTES].[NOME], COUNT(\*) FROM [TABELA DE CLIENTES] LEFT JOIN

[NOTAS FISCAIS] ON [TABELA DE CLIENTES].CPF = [NOTAS FISCAIS].CPF

GROUP BY [TABELA DE CLIENTES].[CPF], [TABELA DE CLIENTES].[NOME]

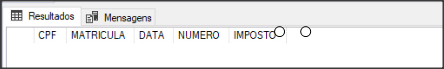
Você terá:



Pegando o CPF que aparece na consulta acima, você pode ver se há alguma nota fiscal associada a ele na tabela de notas fiscais:

SELECT \* FROM [NOTAS FISCAIS] WHERE CPF = '95939180787'

Como esperado, resultado vazio:



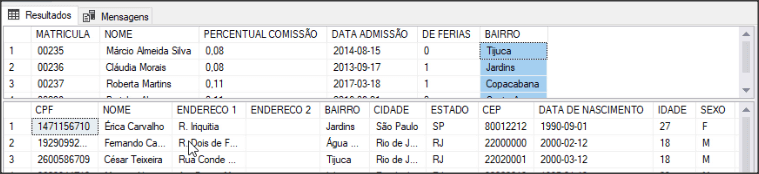
### Usando FULL e CROSS JOIN

16) Abra o aquivo **Usando FULL e CROSS JOIN.sql**.

17) Veja a tabela de vendedores e de clientes:

SELECT \* FROM [TABELA DE VENDEDORES]

SELECT \* FROM [TABELA DE CLIENTES]



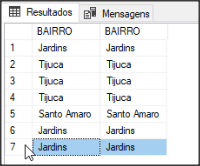
Resultado: 4 vendedores e 15 clientes. O campo em comum entre eles é o **BAIRRO**.

18) Faça o **INNER JOIN** entre a tabela de clientes e de vendedores:

SELECT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE CLIENTES].BAIRRO FROM

[TABELA DE VENDEDORES] INNER JOIN [TABELA DE CLIENTES] ON

[TABELA DE VENDEDORES].BAIRRO = [TABELA DE CLIENTES].BAIRRO



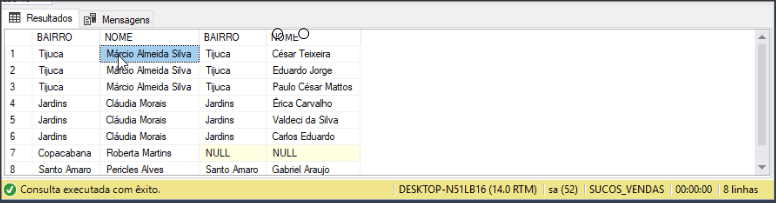
19) Como o resultado foi de 7 linhas, significa que existem vendedores com bairros que não estão na tabela de clientes e vice-versa. O **LEFT JOIN** pode fazer com que você possa descobrir estas não correspondências:

SELECT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME],

[TABELA DE CLIENTES].BAIRRO, [TABELA DE CLIENTES].[NOME] FROM

[TABELA DE VENDEDORES] LEFT JOIN [TABELA DE CLIENTES] ON

[TABELA DE VENDEDORES].BAIRRO = [TABELA DE CLIENTES].BAIRRO



Resultado: 8 linhas

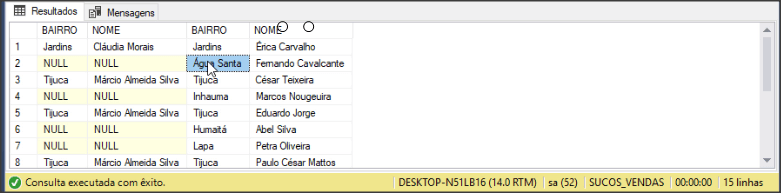
20) Como você ainda não obteve o resultado da maior tabela, significa que existem também vendedores com bairros que não existem na tabela de clientes. Para isso, use o **RIGHT JOIN** para descobrir isto:

SELECT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME],

[TABELA DE CLIENTES].BAIRRO, [TABELA DE CLIENTES].[NOME] FROM

[TABELA DE VENDEDORES] RIGHT JOIN [TABELA DE CLIENTES] ON

[TABELA DE VENDEDORES].BAIRRO = [TABELA DE CLIENTES].BAIRRO



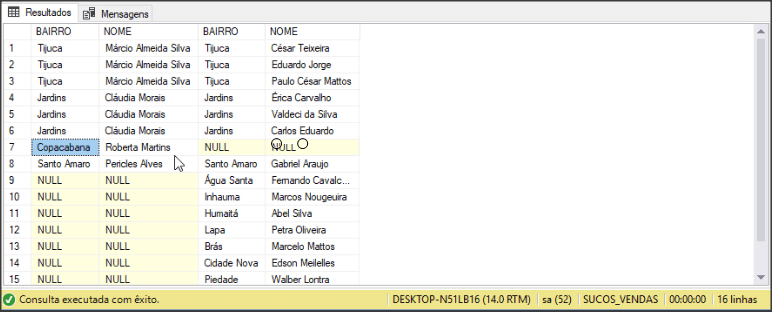
21) Para descobrir, ao mesmo tempo, o cliente que não tem bairro na tabela de vendedores e o vendedor que não tem bairro na tabela de clientes, use o **FULL JOIN**:

SELECT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME],

[TABELA DE CLIENTES].BAIRRO, [TABELA DE CLIENTES].[NOME] FROM

[TABELA DE VENDEDORES] FULL JOIN [TABELA DE CLIENTES] ON

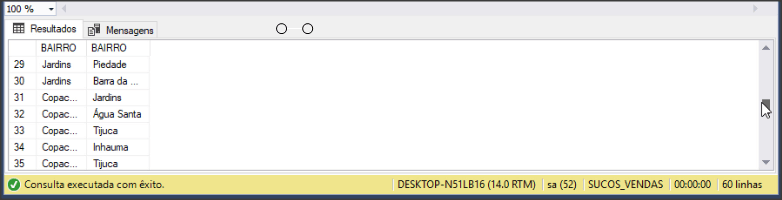
[TABELA DE VENDEDORES].BAIRRO = [TABELA DE CLIENTES].BAIRRO



22) O **CROSS JOIN** irá possibilitar a análise combinatória da tabela de vendedores e clientes:

SELECT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE CLIENTES].BAIRRO FROM

[TABELA DE VENDEDORES] CROSS JOIN [TABELA DE CLIENTES]



O resultado foi justamente 60 linhas, que é são os 15 clientes vezes os 4 vendedores.

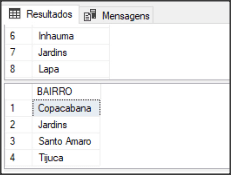
### Juntando consultas

23) Abra o arquivo **Juntando consultas.sql**.

24) Veja a lista de bairros dos clientes e dos vendedores:

SELECT DISTINCT [TABELA DE CLIENTES].BAIRRO FROM [TABELA DE CLIENTES]

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO FROM [TABELA DE VENDEDORES]

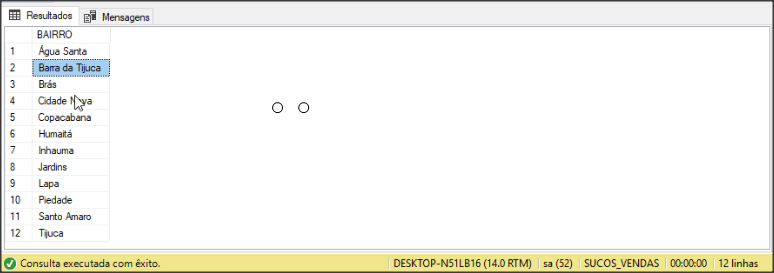


25) Use o **UNION**, juntando essas duas consultas:

SELECT DISTINCT [TABELA DE CLIENTES].BAIRRO FROM [TABELA DE CLIENTES]

UNION

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO FROM [TABELA DE VENDEDORES]



Com o **UNION**, você tem 12 bairros sem que nenhum se repita.

26) Execute agora o **UNION ALL**. Teremos:

SELECT DISTINCT [TABELA DE CLIENTES].BAIRRO FROM [TABELA DE CLIENTES]

UNION ALL

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO FROM [TABELA DE VENDEDORES]



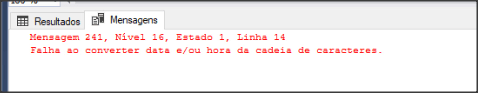
Você terá 15 linhas como resultado, porque os bairros em comum aparecem duplicados.

27) Os campos de um **UNION** devem ter a mesma correspondência de tipo. Veja a consulta abaixo:

SELECT DISTINCT [TABELA DE CLIENTES].[BAIRRO], [TABELA DE CLIENTES].[DATA DE NASCIMENTO] FROM [TABELA DE CLIENTES]

UNION ALL

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME] FROM [TABELA DE VENDEDORES]



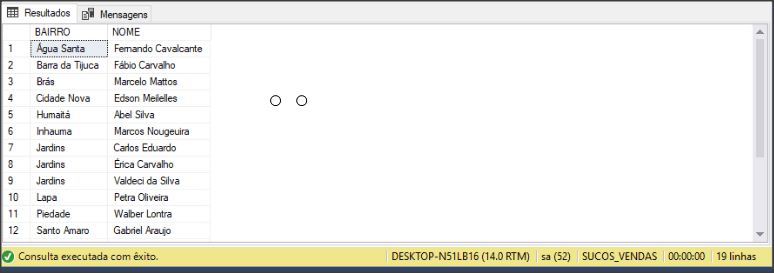
Você terá um erro porque o segundo campo das duas consultas que estão sendo unidas pelo **UNION** são de tipos diferentes.

28) Execute então:

SELECT DISTINCT [TABELA DE CLIENTES].[BAIRRO], [TABELA DE CLIENTES].[NOME] FROM [TABELA DE CLIENTES]

UNION ALL

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME] FROM [TABELA DE VENDEDORES]

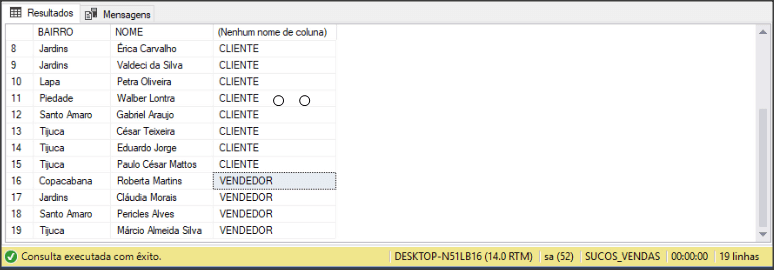


29) O que você pode fazer, em muitos dos casos, é criar uma coluna que identifique a origem do campo quando é efetuado o **UNION**:

SELECT DISTINCT [TABELA DE CLIENTES].[BAIRRO], [TABELA DE CLIENTES].[NOME], 'CLIENTE' FROM [TABELA DE CLIENTES]

UNION ALL

SELECT DISTINCT [TABELA DE VENDEDORES].BAIRRO, [TABELA DE VENDEDORES].[NOME], 'VENDEDOR' FROM [TABELA DE VENDEDORES]

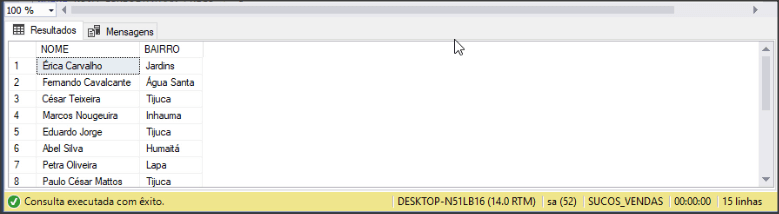


### Sub-consultas

30) Abra o arquivo **Sub consultas.sql**.

31) Veja a consulta de clientes e seus respectivos bairros:

SELECT NOME, BAIRRO FROM [TABELA DE CLIENTES]

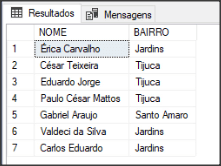


32) Liste somente os clientes que possuam os mesmos bairros que os vendedores. Como você não sabe de antemão quais são os bairros dos vendedores, use a sub-consulta como condição de filtro:

SELECT NOME, BAIRRO FROM [TABELA DE CLIENTES]

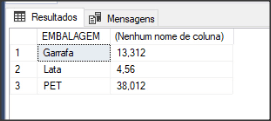
WHERE BAIRRO IN (

SELECT BAIRRO FROM [TABELA DE VENDEDORES])



33) Veja o máximo dos preços por embalagem:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM



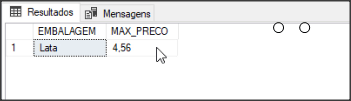
34) Aplique um filtro sobre esta consulta, usando ela como uma sub-consulta no **FROM** de outra consulta:

SELECT NOVA\_CONSULTA.EMBALAGEM, NOVA\_CONSULTA.MAX\_PRECO

FROM (SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) AS MAX\_PRECO FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM)

NOVA\_CONSULTA

WHERE NOVA\_CONSULTA.MAX\_PRECO <= 5



O resultado acima acaba fazendo o mesmo que o **HAVING**.

### Visão

35) Abra o arquivo **Visões.sql**.

36) Note que na consulta abaixo:

SELECT NOVA\_CONSULTA.EMBALAGEM, NOVA\_CONSULTA.MAX\_PRECO

FROM (SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) AS MAX\_PRECO FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM)

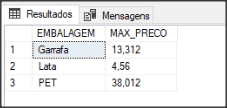
NOVA\_CONSULTA

WHERE NOVA\_CONSULTA.MAX\_PRECO <= 5

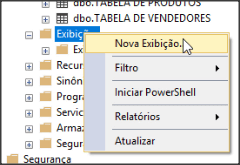
Há uma sub-consulta:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) AS MAX\_PRECO FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM

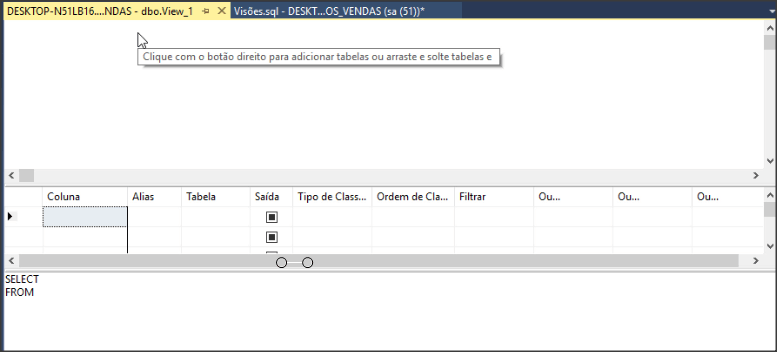
Executando esta sub-consulta, você verá:



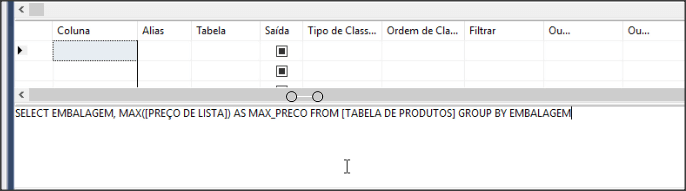
37) Expanda a pasta **Exibições** do banco de dados **SUCOS\_VENDAS**. Clique com o botão da direita do mouse sobre este item e escolha **Nova Exibição**:



38) Serão mostradas as tabelas. Aqui, você pode escolher as tabelas que farão parte das views. Mas, no caso, já há a consulta SQL pronta. Logo, clique em **OK** sem selecionar tabela alguma:



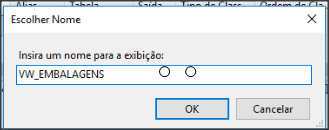
39) onde há escrito **SELECT FROM**, cole a sub-consulta que será usada para a criação da view:



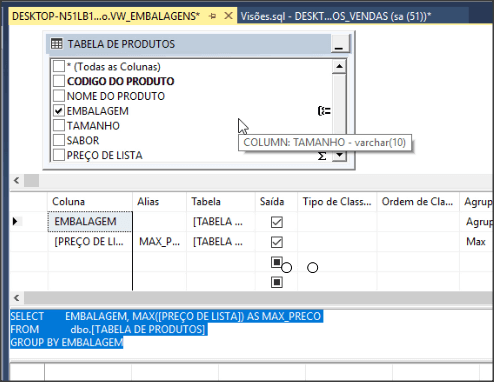
40) Clique no ícone **Salvar** na barra de ferramentas principal:



41) Crie um nome para esta visão (**VW\_EMBALAGENS**):



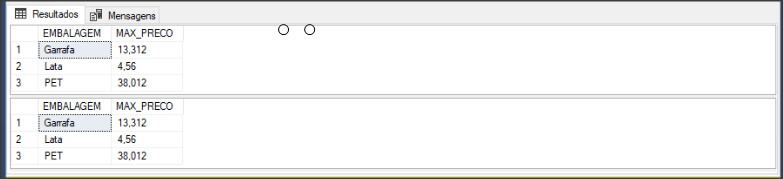
42) Clique em **OK** e você terá a representação gráfica da visão:



43) Agora, use a visão na consulta como se fosse uma tabela. Veja abaixo a seleção original e a visão:

SELECT EMBALAGEM, MAX([PREÇO DE LISTA]) AS MAX\_PRECO FROM [TABELA DE PRODUTOS] GROUP BY EMBALAGEM

SELECT \* FROM [dbo].[VW\_EMBALAGENS]

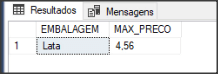


44) Você pode usar a visão como se fosse uma tabela na consulta que, antes, foi usada a sub-consulta:

SELECT NOVA\_CONSULTA.EMBALAGEM, NOVA\_CONSULTA.MAX\_PRECO

FROM VW\_EMBALAGENS NOVA\_CONSULTA

WHERE NOVA\_CONSULTA.MAX\_PRECO <= 5



### Funções de string

1) Abra o arquivo **Consultas string.sql**

2) Execute cada linha e verifique o resultado de cada função:

SELECT LTRIM(' OLA')

SELECT RTRIM('OLA ')

SELECT CONCAT('OLA ','TUDO BEM')

SELECT 'OLA ' + 'TUDO BEM'

SELECT LEFT('RUA AUGUSTA',3)

SELECT RIGHT('RUA AUGUSTA',7)

SELECT UPPER('rua augusta')

SELECT LOWER('RUA AUGUSTA')

SELECT REPLACE('R. AUGUSTA','R.','RUA')

SELECT SUBSTRING('RUA AUGUSTA', 1, 3)

SELECT SUBSTRING('RUA AUGUSTA', 2, 4)

SELECT LEN('RUA AUGUSTA')

SELECT \* FROM [TABELA DE CLIENTES]

SELECT CONCAT(NOME, ' (', CPF, ') ') FROM [TABELA DE CLIENTES]

### Funções de datas

3) Abra o arquivo **Funcoes de datas.sql**.

4) Execute cada linha e verifique o resultado de cada função:

SELECT SYSDATETIME()

SELECT SYSDATETIMEOFFSET()

SELECT SYSUTCDATETIME()

SELECT CURRENT\_TIMESTAMP

SELECT GETDATE()

SELECT GETUTCDATE()

SELECT DATENAME(YEAR,GETDATE())

SELECT DATENAME(MICROSECOND,GETDATE())

SELECT DATENAME(MINUTE,GETDATE())

SELECT DATENAME(MONTH,GETDATE())

SELECT DATEPART(MONTH,GETDATE())

SELECT DAY(GETDATE())

SELECT YEAR(GETDATE())

SELECT DATEFROMPARTS(2015,9,1)

SELECT DATENAME(MONTH,DATEFROMPARTS(2015,9,1))

SELECT DATETIME2FROMPARTS(2015,9,1,13,12,11,120,4)

SELECT DATEDIFF(MONTH, DATEFROMPARTS(2015,9,1), GETDATE())

SELECT DATEADD(MONTH, 5, GETDATE())

SELECT ISDATE('2018-01-01')

SELECT ISDATE('2018-25-28')

SELECT \* FROM [NOTAS FISCAIS]

SELECT [DATA], CONCAT(DATENAME(DAY, [DATA]), ' ', DATENAME(MONTH, [DATA]), ' ', DATENAME(YEAR, [DATA]))

FROM [NOTAS FISCAIS]

### Funções matemáticas

5) Abra o arquivo **Funcoes matemáticas.sql**

6) Execute cada linha e verifique o resultado de cada função:

SELECT CEILING(12.333223)

SELECT FLOOR(12.333223)

SELECT RAND()

SELECT ROUND(12.33323323, 2)

SELECT \* FROM [ITENS NOTAS FISCAIS]

SELECT [QUANTIDADE], [PREÇO], ROUND(([QUANTIDADE] \* [PREÇO]),1) FROM [ITENS NOTAS FISCAIS]

### Conversão de dados

7) Abra o arquivo **Conversao de dados.sql**.

8) Execute cada linha e verifique o resultado de cada função:

SELECT CONVERT(VARCHAR, GETDATE(), 101)

SELECT CONVERT(VARCHAR, GETDATE(), 113)

SELECT CONVERT(VARCHAR, GETDATE(), 130)

SELECT CONVERT(decimal(10,5), 193.57)

SELECT \* FROM [TABELA DE PRODUTOS]

SELECT 'O preço do produto ' + [NOME DO PRODUTO] + ' é ' + [PREÇO DE LISTA]

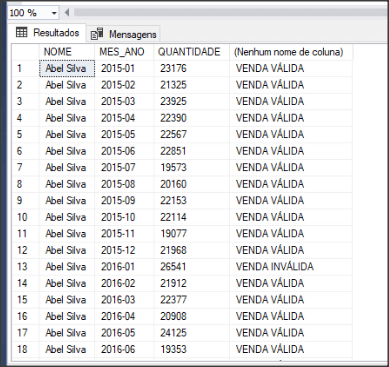
FROM [TABELA DE PRODUTOS]

SELECT 'O preço do produto ' + [NOME DO PRODUTO] + ' é ' + CONVERT(VARCHAR, [PREÇO DE LISTA])

FROM [TABELA DE PRODUTOS]

### Relatório de vendas válidas

1) Objetivo desta prática é a de montar o relatório de vendas por cliente e comparar se o total de venda (quantidade) em um determinado mês superou ou não o limite máximo de venda por cliente (quantidade limite contida no cadastro):



2) Abra o arquivo **Vendas válidas EXEMPLO.sql**.

3) Há 3 tabelas que você irá usar para montar este relatório. A tabela de clientes, para buscar os dados do clientes (nome, por exemplo) e também o valor total de crédito contido no seu cadastro. A outra tabela é a de notas fiscais, onde tem a data da venda. A ligação entre estas duas tabelas é o CPF do cliente. Finalmente, você irá acessar a tabela de itens de nota fiscais, onde você obtém o valor faturado. Esta última tabela irá se juntar com a tabela de notas fiscais pelo número da nota:

SELECT \* FROM [TABELA DE CLIENTES]

SELECT \* FROM [ITENS NOTAS FISCAIS]

SELECT \* FROM [NOTAS FISCAIS]

4) Primeiramente, junte a tabela de notas fiscais com a tabela de itens:

SELECT \* FROM [NOTAS FISCAIS] NF

INNER JOIN [ITENS NOTAS FISCAIS] INF

ON NF.NUMERO = INF.NUMERO



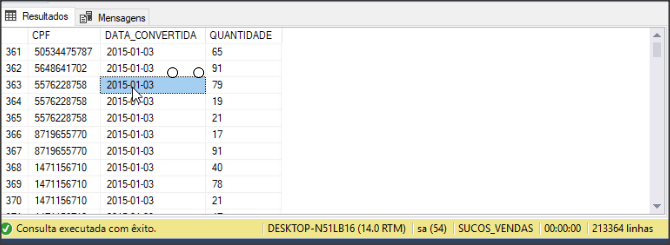
5) A seguir, segregue os campos importantes deste **JOIN**, aplicando algumas funções. Você irá obter apenas o CPF, a data convertida para mês e ano (para ver somente os dados dentro do mês) e a quantidade vendida:

SELECT NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7) AS ANO\_MES,

INF.QUANTIDADE FROM [NOTAS FISCAIS] NF

INNER JOIN [ITENS NOTAS FISCAIS] INF

ON NF.NUMERO = INF.NUMERO



6) Agrupe esta consulta para obter o valor total, por cliente e por mês/ano:

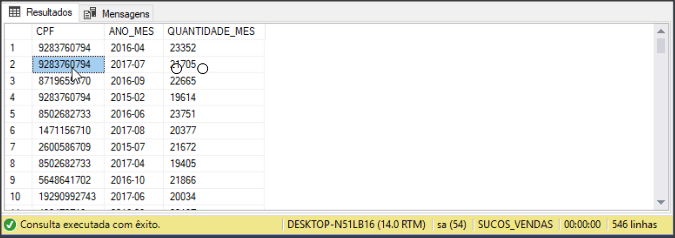
SELECT NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7) AS ANO\_MES,

SUM(INF.QUANTIDADE) AS QUANTIDADE\_MES FROM [NOTAS FISCAIS] NF

INNER JOIN [ITENS NOTAS FISCAIS] INF

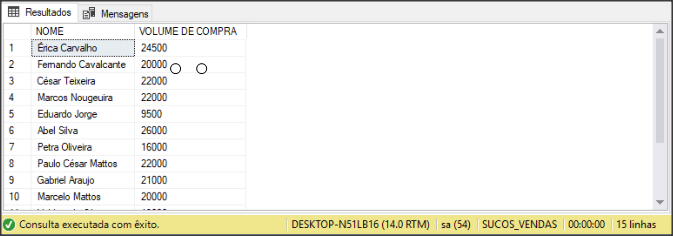
ON NF.NUMERO = INF.NUMERO

GROUP BY NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7)



7) Guarde esta consulta e verifique, agora, os dados do limite de vendas em quantidades contidas na tabela de clientes. Selecione apenas os campos necessários a serem usados no relatório:

SELECT TC.NOME, TC.[VOLUME DE COMPRA] FROM [TABELA DE CLIENTES] TC



8) Junte as duas consultas acima. A que obtém dados de vendas e a que obtém dados do limite máximo:

SELECT TC.NOME, CQ.ANO\_MES, CQ.QUANTIDADE\_MES, TC.[VOLUME DE COMPRA]

FROM

(SELECT NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7) AS ANO\_MES,

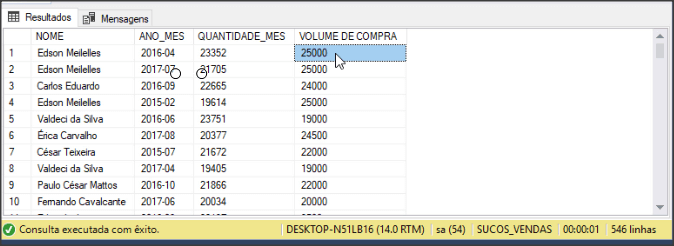
SUM(INF.QUANTIDADE) AS QUANTIDADE\_MES FROM [NOTAS FISCAIS] NF

INNER JOIN [ITENS NOTAS FISCAIS] INF

ON NF.NUMERO = INF.NUMERO

GROUP BY NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7)) CQ

INNER JOIN [TABELA DE CLIENTES] TC ON TC.CPF = CQ.CPF



9) Com duas colunas comparando as vendas com os limites, crie um **CASE** que irá testar se a venda do mês superou ou não a venda limite e escrever um label para isso. Apresente esta consulta ordenada e você terá o relatório final:

SELECT AUX1.NOME, AUX1.ANO\_MES, AUX1.QUANTIDADE\_MES,

CASE WHEN AUX1.QUANTIDADE\_MES <= AUX1.[VOLUME DE COMPRA/10] THEN 'VENDA VÁLIDA'

WHEN AUX1.QUANTIDADE\_MES > AUX1.[VOLUME DE COMPRA/10] THEN 'VENDA INVÁLIDA'

END AS STATUS\_VENDA

FROM

(SELECT TC.NOME, CQ.ANO\_MES, CQ.QUANTIDADE\_MES, TC.[VOLUME DE COMPRA]

FROM

(SELECT NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7) AS ANO\_MES,

SUM(INF.QUANTIDADE) AS QUANTIDADE\_MES FROM [NOTAS FISCAIS] NF

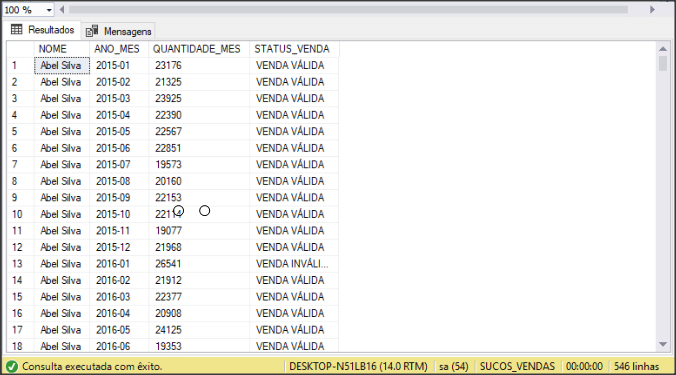
INNER JOIN [ITENS NOTAS FISCAIS] INF

ON NF.NUMERO = INF.NUMERO

GROUP BY NF.CPF, SUBSTRING(CONVERT(VARCHAR, NF.[DATA], 120),1,7)) CQ

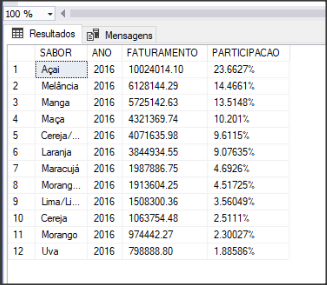
INNER JOIN [TABELA DE CLIENTES] TC ON TC.CPF = CQ.CPF) AUX1

ORDER BY AUX1.NOME, AUX1.ANO\_MES



### Relatório de vendas por sabor

10) O objetivo é montar um relatório de vendas (faturamento) por sabor, dentro de um período de tempo (no caso, no ano de 2016) e criar uma coluna com o percentual de participação de cada venda. É importante exibir os dados ordenados por valor do faturamento, do maior para o menor:



11) Abra o arquivo **Vendas por sabor EXEMPLO.sql**.

12) Esta consulta envolverá a tabela de produtos, para buscar os sabores, a tabela de itens de notas fiscais, para obter o faturamento (quantidade X preço), e a tabela de notas fiscais, onde tem a data da venda. Você irá juntar a tabela de produtos e a tabela de itens de notas fiscais pelo código de produto, e em seguida juntará com a tabela de notas fiscais:

SELECT \* FROM [TABELA DE PRODUTOS] TP

SELECT \* FROM [NOTAS FISCAIS] NF

SELECT \* FROM [ITENS NOTAS FISCAIS] INF

13) Junte as três tabelas, selecionando os campos que são importantes para o relatório:

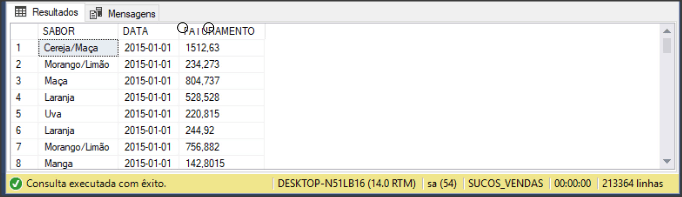
SELECT TP.SABOR, NF.DATA, (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO



14) Agrupe esta consulta por sabor e ano:

SELECT TP.SABOR, YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

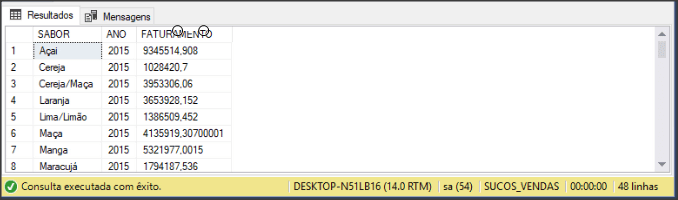
FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

GROUP BY TP.SABOR, YEAR(NF.DATA)



15) Aplique o filtro para ter somente as vendas de 2016:

SELECT TP.SABOR, YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

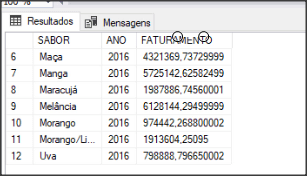
ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY TP.SABOR, YEAR(NF.DATA)



16) Guarde esta consulta e faça outra para obter a soma total de vendas no mesmo período. Para isso, aproveite a consulta anterior, mas tire o sabor do grupo:

SELECT YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

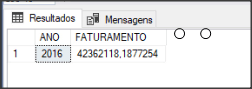
ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY YEAR(NF.DATA)



17) Junte as duas consultas pelo campo que representa o ano:

SELECT AUX1.SABOR, AUX1.ANO, AUX1.FATURAMENTO, AUX2.TOTAL FROM

(SELECT TP.SABOR, YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY TP.SABOR, YEAR(NF.DATA)) AUX1

INNER JOIN (SELECT YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS TOTAL

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

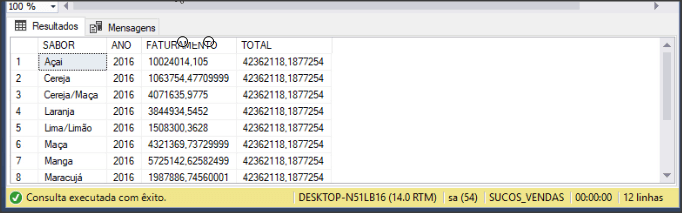
INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY YEAR(NF.DATA)) AUX2

ON AUX1.ANO = AUX2.ANO



18) Para encerrar o relatório, calcule a coluna percentual e ordene a saída do maior para o menor faturamento. Aproveite e converta os dados de faturamento e percentual para ser representado com duas casas decimais:

SELECT AUX1.SABOR, AUX1.ANO, CONVERT(DECIMAL(15,2), AUX1.FATURAMENTO) AS FATURAMENTO,

CONVERT(VARCHAR, CONVERT(DECIMAL(15,2),(AUX1.FATURAMENTO/AUX2.TOTAL) \* 100)) + ' %'

AS PERCENTUAL FROM

(SELECT TP.SABOR, YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS FATURAMENTO

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY TP.SABOR, YEAR(NF.DATA)) AUX1

INNER JOIN (SELECT YEAR(NF.DATA) AS ANO, SUM (INF.QUANTIDADE \* INF.PREÇO) AS TOTAL

FROM [ITENS NOTAS FISCAIS] INF INNER JOIN [TABELA DE PRODUTOS] TP

ON TP.[CODIGO DO PRODUTO] = INF.[CODIGO DO PRODUTO]

INNER JOIN [NOTAS FISCAIS] NF

ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA) = 2016

GROUP BY YEAR(NF.DATA)) AUX2

ON AUX1.ANO = AUX2.ANO

ORDER BY AUX1.FATURAMENTO DESC

