Nesse momento, já somos capazes de colocar em prática todo o nosso conhecimento de SQL relativo às consultas. Vamos revisar e adaptar tudo que aprendemos neste curso para atender a um pedido do gerente de vendas da empresa de sucos: ele solicitou um relatório para evidenciar, dentre todas as vendas realizadas, quais foram válidas e quais foram inválidas.

Primeiramente, vamos esclarecer o que é uma **venda válida**. Na tabela de clientes, há uma coluna chamada "VOLUME\_DE\_COMPRA" que corresponde ao volume máximo de suco que cada cliente pode comprar em um mês. Para saber se esse número foi respeitado, podemos consultar a tabela de itens de notas fiscais e compará-lo à quantidade adquirida por cada cliente em cada mês. Vendas válidas são aquelas que não extrapolam esse limite **mensal**.

Então, devemos entregar um relatório sobre vendas válidas e inválidas ao gerente de vendas para que possa ser analisado se o parâmetro de volume de compras está sendo considerado durante as negociações com os clientes.

Agora que compreendemos nossa meta, aos poucos vamos construir uma seleção que atenda essa demanda. Começaremos abrindo o MySQL Workbench e criando um script. Nosso ***primeiro passo*** será descobrir o volume total de vendas para cada cliente dentro de um mês, lembrando que o mês é o fator limitante.

Vamos consultar a tabela de itens das notas fiscais e a tabela de notas fiscais para saber o que cada uma delas pode nos mostrar:

Texto

O conteúdo gerado por IA pode estar incorreto.

As informações que nos interessam estão em tabelas diferentes: os dados sobre o cliente (**CPF**) e a **data da venda** estão na tabela de notas fiscais, enquanto a **quantidade** das vendas consta na tabela de itens das notas fiscais. Para unificar essa visualização, vamos fazer uma seleção com INNER JOIN:

Texto

O conteúdo gerado por IA pode estar incorreto.

Note que criamos um *alias* para cada tabela ("NF" e "INF") e usamos o número das notas fiscais como campo em comum para realizar essa junção. Como usamos o asterisco, o retorno mostrará **todas** as colunas de ambas as tabelas, mas não precisamos de todas essas informações. Então, vamos refazer essa consulta, selecionando apenas as colunas relevantes para nosso relatório:

Texto

O conteúdo gerado por IA pode estar incorreto.

Agora, vejamos o formato das datas. Tendo em vista nosso objetivo, seria interessante agrupar as vendas **mensalmente**, porém no momento as datas na coluna "DATA\_VENDA" são exibidas dia a dia. Para trocar esse formato, usaremos a função DATE\_FORMAT():

Texto

O conteúdo gerado por IA pode estar incorreto.

Dessa forma, usamos os **especificadores** %Y e %m para exibir somente o ano e o mês de cada venda. Ademais, renomeamos o campo "DATA\_VENDA" para "MES\_ANO".

Analisando o retorno, há vários registros cujo CPF e a data são iguais. Os quatro primeiros registros, por exemplo, são vendas realizadas em janeiro de 2015 para o cliente de CPF 7771579779. Se temos vendas no **mesmo mês** para o **mesmo cliente**, podemos agrupá-las por esses campos:

**SELECT** NF.CPF, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m') **AS** MES\_ANO, SUM(INF.QUANTIDADE) **AS** QUANTIDADE\_VENDAS **FROM** NOTAS\_FISCAIS NF

**INNER** **JOIN** ITENS\_NOTAS\_FISCAIS INF

**ON** NF.NUMERO = INF.NUMERO

**GROUP** **BY** NF.CPF, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m');

Note que, no GROUP BY, inserimos a função DATE\_FORMAT() inteira e não somente o campo "NF.DATA\_VENDA". Nem sempre essa inclusão é necessária, dependerá do resultado da função, mas para garantir o sucesso, colocamos a função completa (sem o *alias*!).

Com essa seleção, além de agrupar registros por CPF e data, também aplicamos SUM() aos valores da coluna "QUANTIDADE" e a esse total demos o nome de "QUANTIDADE\_VENDAS". Assim, o retorno será uma lista em que cada registro corresponde à quantidade de vendas para um determinado cliente em um determinado mês. Em outras palavras, concluímos nosso ***primeiro passo***.

Antes de continuar, vamos fazer um **comentário** no script para não nos esquecer dessa seleção. Na linha acima da consulta, vamos digitar o seguinte texto:

/\* CONSULTA COM VENDAS DE CLIENTES POR MES \*/

O comentário é um texto que não altera a execução do código, o programa irá ignorar esse trecho quando estiver rodando. No SQL, basta colocar os símbolos /\* no início, e \*/ ao final. Quanto maior e mais complexo nosso script ficar, mais importante é colocar comentários para lembrarmos o que cada parte faz.

O ***segundo passo*** é mais simples: exibir o volume de compra permitido para cada cliente. Encontraremos essa informação na tabela de clientes, à qual daremos o *alias* "TC":



Novamente, temos uma consulta que retorna mais colunas do que necessitamos. Vamos selecionar apenas os campos relevantes:

Texto

O conteúdo gerado por IA pode estar incorreto.

Assim, obtemos o CPF, o nome e o volume de compra de cada cliente — terminamos o ***segundo passo***. Podemos também fazer um comentário sobre essa consulta para nos lembrarmos dela:

Texto

O conteúdo gerado por IA pode estar incorreto.

Por fim, o ***terceiro passo*** será comparar as tabelas resultantes do primeiro e do segundo passo. Faremos um INNER JOIN entre elas:

**SELECT** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m') **AS** MES\_ANO, SUM(INF.QUANTIDADE) **AS** QUANTIDADE\_VENDAS, TC.VOLUME\_DE\_COMPRA **AS** QUANTIDADE\_LIMITE **FROM** NOTAS\_FISCAIS NF

**INNER** **JOIN** ITENS\_NOTAS\_FISCAIS INF

**ON** NF.NUMERO = INF.NUMERO

**INNER** **JOIN** TABELA\_DE\_CLIENTES TC

**ON** TC.CPF = NF.CPF

**GROUP** **BY** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m');

Assim, fizemos um JOIN cujo campo em comum é o CPF. Além do mais, incluímos as colunas "TC.NOME" e "TC.VOLUME\_DE\_COMPRA" no início do SELECT para serem exibidas no retorno. Também inserimos "TC.NOME" no GROUP BY ao final da seleção.

Ao tentar executar, receberemos um erro. O problema está no trecho TC.VOLUME\_DE\_COMPRA. Quando agrupamos registros, somos obrigados a especificar o que deve ser feito com os campos numéricos. Por exemplo, em SUM(INF.QUANTIDADE) temos a indicação de que os valores devem ser somados. Já em TC.VOLUME\_DE\_COMPRA não há especificação. Uma solução, então, é usar a função MAX() para que a coluna exiba o maior valor:

**SELECT** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m') **AS** MES\_ANO, SUM(INF.QUANTIDADE) **AS** QUANTIDADE\_VENDAS, MAX(TC.VOLUME\_DE\_COMPRA) **AS** QUANTIDADE\_LIMITE **FROM** NOTAS\_FISCAIS NF

**INNER** **JOIN** ITENS\_NOTAS\_FISCAIS INF

**ON** NF.NUMERO = INF.NUMERO

**INNER** **JOIN** TABELA\_DE\_CLIENTES TC

**ON** TC.CPF = NF.CPF

**GROUP** **BY** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m');

Note que também poderíamos usar MIN() já que o volume de compra é constante para cada CPF - ou seja, nesse caso, o valor máximo e o mínimo são o mesmo. O uso de SUM(), no entanto, não funcionaria, pois os valores seriam somados e o volume permitido aumentaria.

Com essa consulta, cada registro exibirá o CPF e o nome do cliente, o mês e o ano das vendas, a soma das quantidades adquiridas no mês em questão e o volume de compras permitido. O primeiro registro, por exemplo, informa que em janeiro de 2015, Érica Carvalho comprou 24316 sucos do seu volume máximo de 24500 — está abaixo de seu limite, ou seja, é uma venda válida. Já no quinto registro, sabemos que em maio do mesmo ano ela extrapolou o volume permitido (26385 de 24500), então trata-se de uma venda inválida.

Em outras palavras, já somos capazes de averiguar se cada venda foi válida ou não, ao subtrair a quantidade limite da quantidade de vendas. Mas não convém enviar o relatório do jeito que está agora, pois o gerente teria que calcular os valores linha a linha. Em vez disso, criaremos um script que faça essa operação por nós e automaticamente mostre se é uma venda válida ou não.

Vamos usar a última seleção como uma subconsulta com *alias* "X". Em seguida, incluiremos uma coluna chamada "DIFERENCA" que exiba o resultado de X.QUANTIDADE\_LIMITE - X.QUANTIDADE\_VENDAS:

**SELECT** X.CPF, X.NOME, X.MES\_ANO, X.QUANTIDADE\_VENDAS, X.QUANTIDADE\_LIMITE,

X.QUANTIDADE\_LIMITE - X.QUANTIDADE\_VENDAS **AS** DIFERENCA

**FROM** (

**SELECT** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m') AS MES\_ANO,

SUM(INF.QUANTIDADE) **AS** QUANTIDADE\_VENDAS , MAX(TC.VOLUME\_DE\_COMPRA) **AS** QUANTIDADE\_LIMITE **FROM** NOTAS\_FISCAIS NF

INNER **JOIN** ITENS\_NOTAS\_FISCAIS INF

**ON** NF.NUMERO = INF.NUMERO

INNER **JOIN** TABELA\_DE\_CLIENTES TC

**ON** TC.CPF = NF.CPF

**GROUP** **BY** NF.CPF, TC.NOME, DATE\_FORMAT(NF.DATA\_VENDA, '%Y-%m')) X;

Depois, para facilitar a leitura desses dados, vamos acrescentar uma coluna que apresente o texto "válida" ou "inválida" conforme o resultado da coluna "DIFERENCA". **Quando** negativo, a venda é inválida; **senão**, é válida:

Texto

O conteúdo gerado por IA pode estar incorreto.

Agora que temos a coluna "STATUS\_VENDA" nos informando se cada venda é válida ou não, o campo "DIFERENCA" se torna desnecessário, então podemos retirá-lo da consulta:

Texto

O conteúdo gerado por IA pode estar incorreto.

Essa seleção nos permitirá analisar todas as vendas mensais por CPF e rapidamente avaliar quais delas estavam dentro do limite de cada cliente. Por exemplo, veremos que Fernando Cavalcante realizou várias compras inválidas mas, ao mesmo tempo, ele tinha um limite menor que o de Érica Carvalho.

Desse modo, conseguimos reunir em uma consulta todos os dados pertinentes para o relatório que foi solicitado pelo gerente de vendas da empresa de sucos.