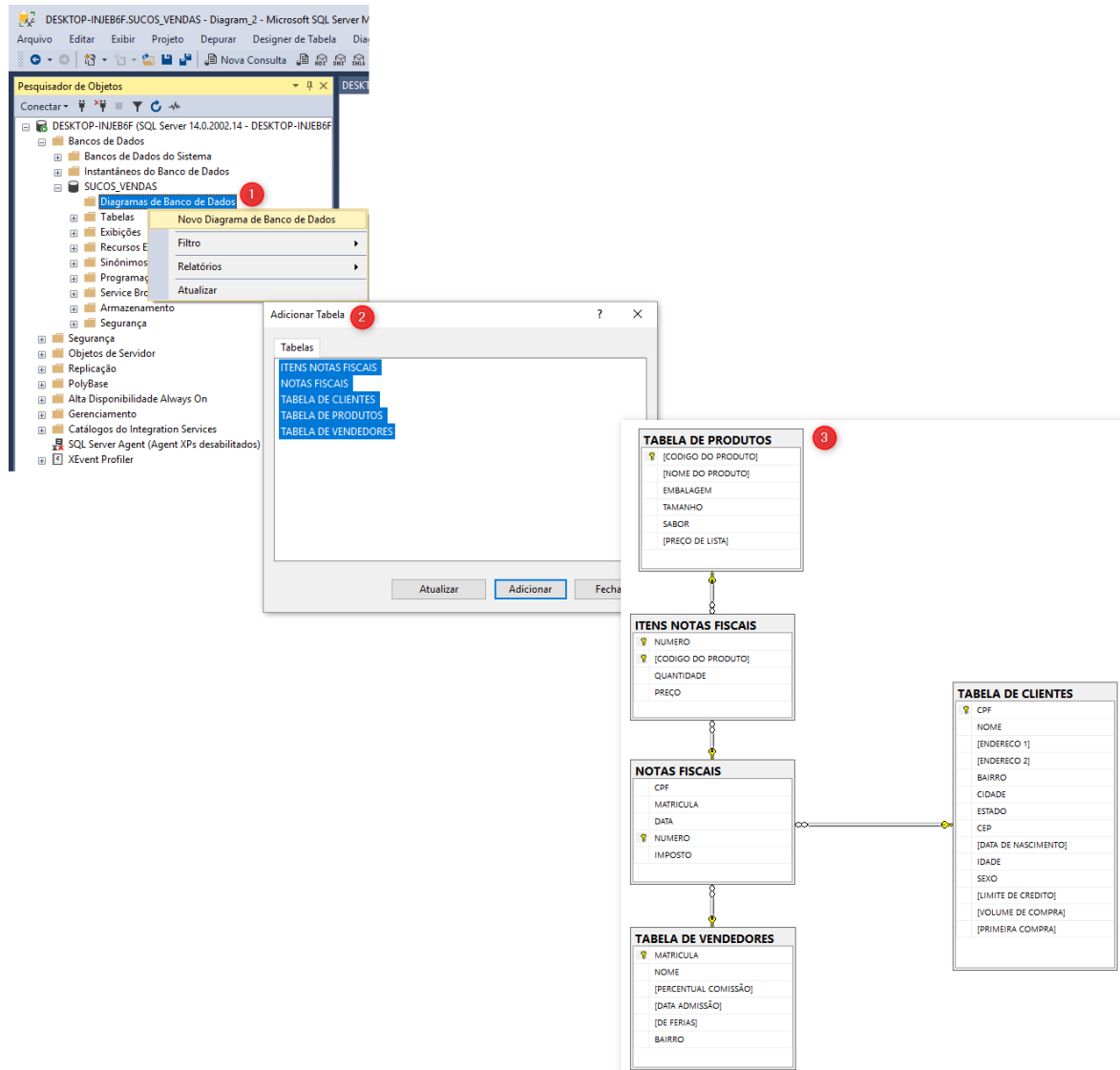


# SQL Server - Consultas Inteligentes

## 1 - CRIAR DIAGRAMA DE RELACIONAMENTO

Conhecer o diagrama de classes é fundamental para saber como realizar as consultas.



## 2 - FUNÇÕES AGREGADAS

- COUNT(): Apenas conta os valores
- SUM(): Soma dos valores
- MAX(): Maior valor
- MIN(): Menor valor
- AVG(): Média dos valores

```
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
```

### 3 - UTILIZANDO O CASE EM PROJEÇÕES

```
SELECT
    [NOME],
    CASE
        WHEN YEAR([DATA DE NASCIMENTO]) < 1990 THEN 'ADULTO'
        WHEN YEAR([DATA DE NASCIMENTO]) >= 1990 AND YEAR([DATA DE NASCIMENTO]) <= 1995 THEN 'JOVEM'
        ELSE 'CRIANÇA'
    END 'CLASSIFICAÇÃO'
FROM [TABELA DE CLIENTES]
```

### 4 - JOINS

Relacionar as tabelas de acordo com os valores (geralmente CHAVE PRIMÁRIA).

**OBS:** a junção só pode ocorrer com atributos com **mesmo tipo**.

- **INNER JOIN:** Junta APENAS os registros que existem nas duas tabelas.
- **LEFT JOIN / RIGHT JOIN:** Faz a junção mesmo que NÃO exista os registros de junção nas duas tabelas (ESQUERDA / DIREITA DA TABELA).

```
SELECT [NOME], COUNT(N.[NUMERO])
FROM [TABELA DE CLIENTES] C
LEFT JOIN [NOTAS FISCAIS] N ON N.CPF = C.CPF
GROUP BY [NOME]
```

No exemplo acima:

- **LEITURA:** Preciso trazer TODOS os clientes, mesmo que não tenha nenhum registro desta tabela que está à ESQUERDA na tabela da direita.
- **OBS:** Ao realizar COUNT(), atentar-se ao atributo dentro da função: o que eu devo considerar para somar. No exemplo acima, devo considerar a PK da Nota Fiscal, uma vez que se um cliente não tiver nenhuma nota o COUNT me retornará ZERO para este cliente.

- **FULL JOIN: RIGHT JOIN + LEFT JOIN** entre as tabelas. Quando não houver junção irá trazer NULL nos atributos tanto de uma tabela quanto da outra.

**Query 1: RIGHT JOIN**

```
SELECT C.[BAIRRO] AS 'BAIRRO CLIENTE', V.[BAIRRO] AS 'BAIRRO VEDEDOR'
FROM [TABELA DE CLIENTES] C
RIGHT JOIN [TABELA DE VENDEDORES] V ON V.BAIRRO = C.BAIRRO
```

BAIRRO CLIENTE	BAIRRO VEDEDOR
Tijuca	Tijuca
Tijuca	Tijuca
Tijuca	Tijuca
Jardins	Jardins
Jardins	Jardins
Jardins	Jardins
NULL	Copacabana
Santo Amaro	Santo Amaro

**Query 2: LEFT JOIN**

```
SELECT C.[BAIRRO] AS 'BAIRRO CLIENTE', V.[BAIRRO] AS 'BAIRRO VEDEDOR'
FROM [TABELA DE CLIENTES] C
LEFT JOIN [TABELA DE VENDEDORES] V ON V.BAIRRO = C.BAIRRO
```

BAIRRO CLIENTE	BAIRRO VEDEDOR
Jardins	Jardins
Água Santa	NULL
Tijuca	Tijuca
Inhauma	NULL
Tijuca	Tijuca
Humatã	NULL
Lapa	NULL
Tijuca	Tijuca
Santo Amaro	Santo Amaro
Brás	NULL
Jardins	Jardins
Jardins	Jardins
Cidade Nova	NULL
Piedade	NULL
Barra da Tijuca	NULL

**Query 3: FULL JOIN**

```
SELECT C.[BAIRRO] AS 'BAIRRO CLIENTE', V.[BAIRRO] AS 'BAIRRO VEDEDOR'
FROM [TABELA DE CLIENTES] C
FULL JOIN [TABELA DE VENDEDORES] V ON V.BAIRRO = C.BAIRRO
```

BAIRRO CLIENTE	BAIRRO VEDEDOR
Jardins	Jardins
Água Santa	NULL
Tijuca	Tijuca
Inhauma	NULL
Tijuca	Tijuca
Humatã	NULL
Lapa	NULL
Tijuca	Tijuca
Santo Amaro	Santo Amaro
Brás	NULL
Jardins	Jardins
Jardins	Jardins
Cidade Nova	NULL
Piedade	NULL
Barra da Tijuca	NULL
NULL	Copacabana

- **CROSS JOIN:** Todas as combinações possíveis entre DUAS tabelas.

#### 4.1 - JUNTANDO CONSULTAS

As consultas são operações de **Álgebra Relacional**: na teoria de conjuntos utilizar a UNIÃO, INTERSECÇÃO, EXCEÇÃO(DIFERENÇA) E O PRODUTO CARTESIANO (CROSS JOIN). Estas operações só podem ser utilizadas se:

- A **projeção** for **igual** em ambas as consultas;
- O **tipo** em cada **atributo** da projeção for **igual**.

##### UNION / UNION ALL

- **UNION:** Faz a união de 2 consultas / *conjunto* em um resultado, **aplicando o DISTINCT**.
- **UNION ALL:** Faz a união de 2 consultas / *conjunto* em um resultado, **NÃO aplicando o DISTINCT**.

**INTERSECT:** Faz a intersecção entre 2 consultas / *conjuntos* em um resultado. Apresenta apenas o que existe nos DOIS conjuntos.

**EXCEPT:** Mostra o que existe apenas no PRIMEIRO conjunto. **Exclui** os registros que existem no SEGUNDA conjunto para exibir na consulta.

**OUTER APPLY:** Permite realizar uma outra consulta a partir da primeira.

```
-- Representante da Propostas_Vendas_Testes DIFERENTE do Representante em Propostas_X_Vendedores_Testes

SELECT P.CodigoProposta
FROM Propostas_Vendas_Testes P
OUTER APPLY(
    SELECT PV1.VendedorId
    FROM Propostas_X_Vendedores_Testes PV1
    WHERE PV1.CodigoProposta = P.CodigoProposta
    AND PV1.CodigoTipoVendedor = 1
) AS PTV
WHERE P.RepresentanteId <> PTV.VendedorId
```

É realizado uma OUTRA  
BUSCA para CADA registro  
em  
'Propostas\_Vendas\_Testes'

Condição que faz apresentar somente os  
casos em que o Representante em  
'Propostas\_Vendas\_Testes' seja diferente  
do vendedor representante em  
'Propostas\_X\_Vendedores\_Testes'

## 4.2 - SUB-CONSULTAS

Faz o papel de uma tabela temporária dentro de um **IN / FROM**.

### 1 Sub-consulta dentro do IN: Buscar por campo em outro SELECT

```
SELECT *
FROM [TABELA DE VENDEDORES]
WHERE NOME IN(SELECT Nome FROM [TABELA DE CLIENTES])
```

	MATRICULA	NOME	PERCENTUAL COMISSÃO	DATA ADMISSÃO	DE FERIAS	BAIRRO
1	5	Abel Silva	20	2018-12-06	0	ITAPARK

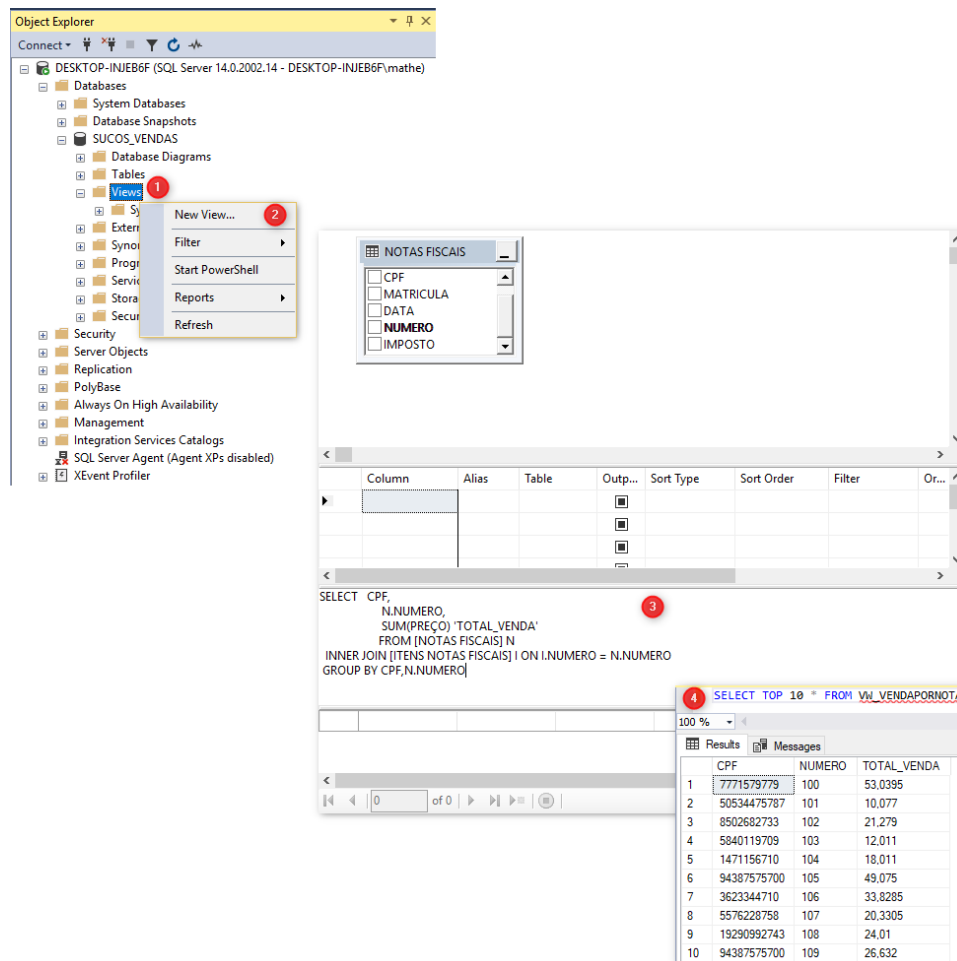
### 2 Sub-consulta no FROM: Criação de tabela temporária para realizar as determinadas junções sem a necessidade de criar estas temporária num INTO

```
SELECT V.Nome, V.Matricula, C.Nome
FROM (SELECT Matricula, Nome
      FROM [TABELA DE VENDEDORES]) V
INNER JOIN [TABELA DE CLIENTES] C ON C.NOME = V.NOME
```

	Nome	Matricula	Nome
1	Abel Silva	5	Abel Silva

## 4.3 - VIEWS

Consulta que é criada e armazenada.

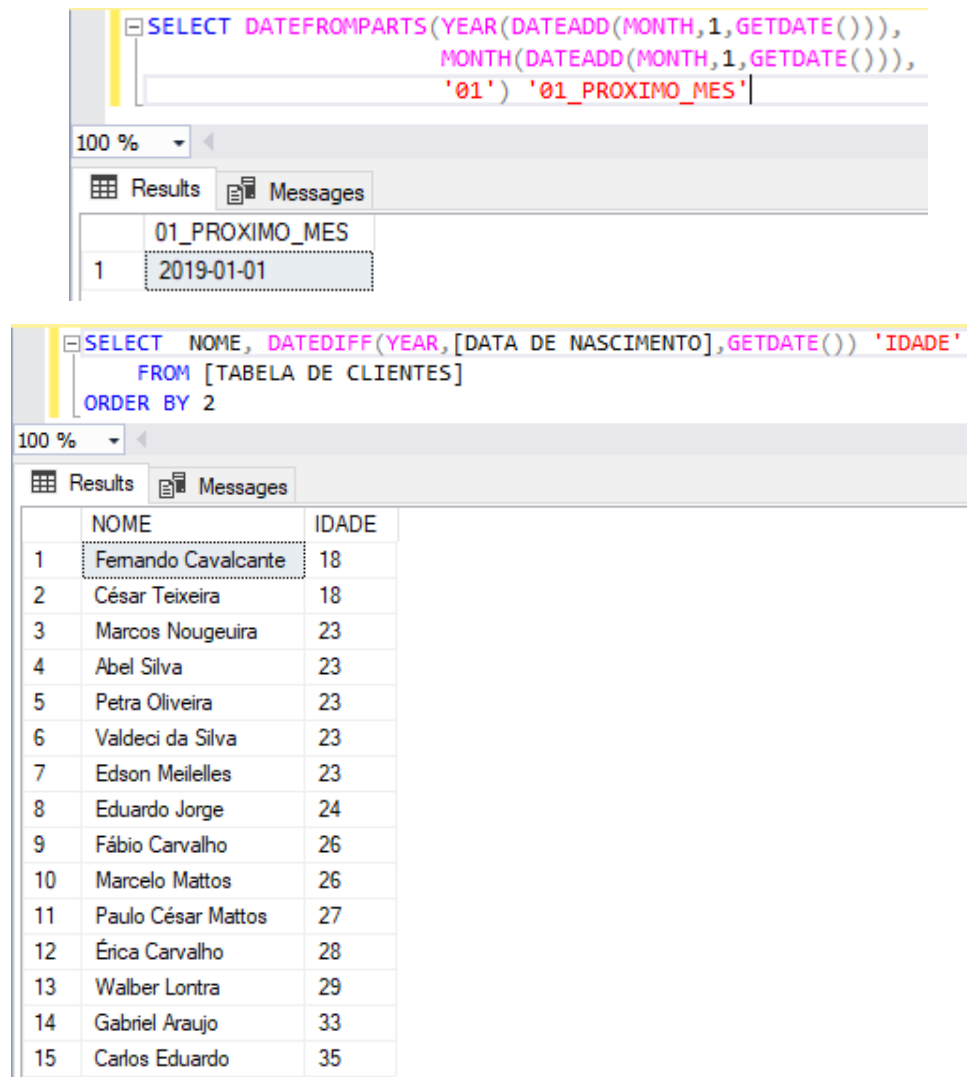


## 5 - FUNÇÕES DO SQL SERVER

### 5.1 - Funções de STRING

- **TRIM(expression):** Remove TODOS espaços em branco.
  - **LTRIM(expression):** Remove os espaços em branco À ESQUERDA
  - **RTRIM(expression):** Remove os espaços em branco À DIREITA
- **SUBSTRING(expression, start, length):** Retorna nova *string* a partir de uma expressão, o índice de início e o tamanho desejado
  - **LEFT (character, length):** Retorna nova *string*, a contar pela esquerda
  - **RIGHT (character, length):** Retorna nova *string*, a contar pela direita
- **LEN(expression):** Retorna o tamanho de uma expressão
- **UPPER(expression):** Retorna nova *string*, com letras em Maiúsculo
- **LOWER(expression):** Retorna nova *string*, com letras em Minúsculo
- **REPLACE(expression, old\_char, new\_char):** Retorna nova *string*, alterando os caracteres desejados
- **CONCAT(expression[,...]):** Concatena uma série de *strings* = 'A' + 'B' ('AB') em um único campo.

## 5.2 - Funções de DATA



**Query 1:**

```
SELECT DATEFROMPARTS(YEAR(DATEADD(MONTH,1,GETDATE())),
                      MONTH(DATEADD(MONTH,1,GETDATE())),
                      '01') '01_PROXIMO_MES'
```

	01_PROXIMO_MES
1	2019-01-01

**Query 2:**

```
SELECT NOME, DATEDIFF(YEAR,[DATA DE NASCIMENTO],GETDATE()) 'IDADE'
FROM [TABELA DE CLIENTES]
ORDER BY 2
```

	NOME	IDADE
1	Fernando Cavalcante	18
2	César Teixeira	18
3	Marcos Nogueira	23
4	Abel Silva	23
5	Petra Oliveira	23
6	Valdeci da Silva	23
7	Edson Meilless	23
8	Eduardo Jorge	24
9	Fábio Carvalho	26
10	Marcelo Mattos	26
11	Paulo César Mattos	27
12	Érica Carvalho	28
13	Walber Lontra	29
14	Gabriel Araujo	33
15	Carlos Eduardo	35

**OBS:** Arquivo em anexo na pasta com funções e explicações.

## 5.3 - Funções MATEMÁTICAS

- **CEILING(expression):** Arredonda para MAIOR inteiro
- **FLOOR(expression):** Arredonda para MENOR inteiro
- **ROUND(expression, length):** Arredonda de acordo com a expressão, levando em consideração o valor **mais próximo**. Recebe a **quantidade de casas decimais** desejadas.

```
SELECT CEILING(12.01) -- 13
SELECT FLOOR(12.99) -- 12
SELECT ROUND(12.49,0) -- 12
SELECT ROUND(12.5,0) -- 13
SELECT ROUND(12.365,2) -- 12.37
```

## 5.4 - Conversão de Dados

**CAST:** É padrão do SQL, existe no ORACLE/MYSQL.

**CONVERT:** Específico do SQL Server. Possui funcionalidades mais complexas.

<https://docs.microsoft.com/pt-br/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-2017>

- CONVERT(<TIPO>,<CAMPO>,[<ESTILO (PARA DATA)>])

```
SELECT CONVERT(VARCHAR, GETDATE(), 101) -- 12/10/2018
SELECT CONVERT(DECIMAL(10,5), 193.57) -- 193.57
SELECT 'O preço do produto ' + [NOME DO PRODUTO] + ' é ' + CONVERT(VARCHAR, [PREÇO DE LISTA])
FROM [TABELA DE PRODUTOS] -- CONVERTER INT EM VARCHAR PARA CONCATENÁ-LO
```

```
SELECT 'O cliente ' + C.NOME + ' faturou ' + CONVERT(VARCHAR, CONVERT(DECIMAL(15,2), SUM((I.QUANTIDADE * I.PREÇO) * N.IMPOSTO)))
FROM [TABELA DE CLIENTES] C
INNER JOIN [NOTAS FISCAIS] N ON N.CPF = C.CPF
INNER JOIN [ITENS NOTAS FISCAIS] I ON I.NUMERO = N.NUMERO
WHERE YEAR(N.DATA) = '2016'
GROUP BY C.NOME
```

100 %

Results Messages

	(No column name)
1	O cliente Abel Silva faturou 340840.57
2	O cliente Carlos Eduardo faturou 310755.24
3	O cliente Céear Teixeira faturou 326886.73
4	O cliente Edson Meilletes faturou 337261.67
5	O cliente Eduardo Jorge faturou 337534.97

## 6 – EXEMPLOS DE RELATÓRIOS

### 6.1 - NOME, ANO/MÊS DA VENDA, QUANTIDADE DE PRODUTOS NO MÊS, STATUS DA VENDA (VÁLIDO SE VOLUME DE COMPRA MAIOR OU IGUAL À QUANTIDADE)

```
-- NOME , ANO MES, QTD, STATUS VENDA (VÁLIDA OU INVÁLIDA, DE ACORDO COM O VOLUME DE COMPRA)

SELECT AUX_1.NOME,
       AUX_1.DATA,
       AUX_1.QUANTIDADE,
       CASE
         WHEN AUX_1.[VOLUME DE COMPRA] < AUX_1.[QUANTIDADE] THEN 'INVÁLIDO'
         ELSE 'VÁLIDO'
       END AS 'STATUS_VENDA'
FROM (SELECT C.NOME,
            SUBSTRING(CONVERT(VARCHAR,N.[DATA],120),1,7) 'DATA',
            C.[VOLUME DE COMPRA],
            SUM(I.QUANTIDADE) 'QUANTIDADE'
      FROM [TABELA DE CLIENTES] C
      INNER JOIN [NOTAS FISCAIS] N ON N.[CPF] = C.[CPF]
      INNER JOIN [ITENS NOTAS FISCAIS] I ON I.[NUMERO] = N.[NUMERO]
     GROUP BY C.NOME,
            SUBSTRING(CONVERT(VARCHAR,N.[DATA],120),1,7),
            C.[VOLUME DE COMPRA]) AS AUX_1
ORDER BY 1 ASC, 2 ASC
```

	NOME	DATA	QUANTIDADE	STATUS_VENDA
1	Abel Silva	2015-01	23176	VÁLIDO
2	Abel Silva	2015-02	21325	VÁLIDO
3	Abel Silva	2015-03	23925	VÁLIDO
4	Abel Silva	2015-04	22390	VÁLIDO
5	Abel Silva	2015-05	22567	VÁLIDO

### 6.2 - BUSCAR SOMENTE INVÁLIDOS E EXIBIR PERCENTUAL ULTRAPASSO NA QUANTIDADE

```
-- NOME , ANO MES, QTD, PERCENTUAL ULTRASSADO DE ACORDO COM O LIMITE DE VOLUME

SELECT AUX_1.NOME,
       AUX_1.DATA,
       AUX_1.QUANTIDADE,
       ROUND((((AUX_1.[QUANTIDADE] - AUX_1.[VOLUME DE COMPRA]) * 100) / AUX_1.[VOLUME DE COMPRA]),2) AS 'PERCENTUAL ULTRAPASSADO'
FROM (SELECT C.NOME,
            SUBSTRING(CONVERT(VARCHAR,N.[DATA],120),1,7) 'DATA',
            C.[VOLUME DE COMPRA],
            SUM(I.QUANTIDADE) 'QUANTIDADE'
      FROM [TABELA DE CLIENTES] C
      INNER JOIN [NOTAS FISCAIS] N ON N.[CPF] = C.[CPF]
      INNER JOIN [ITENS NOTAS FISCAIS] I ON I.[NUMERO] = N.[NUMERO]
     GROUP BY C.NOME,
            SUBSTRING(CONVERT(VARCHAR,N.[DATA],120),1,7),
            C.[VOLUME DE COMPRA]) AS AUX_1
WHERE AUX_1.[VOLUME DE COMPRA] < AUX_1.[QUANTIDADE]
ORDER BY 1 ASC, 2 ASC
```

	NOME	DATA	QUANTIDADE	PERCENTUAL ULTRAPASSADO
1	Abel Silva	2016-01	26541	2.08
2	Carlos Eduardo	2015-07	24904	3.77
3	César Teixeira	2015-02	22949	4.31
4	César Teixeira	2015-05	24722	12.37
5	César Teixeira	2015-06	23158	5.26



### 6.3 - SABOR, ANO, FATURAMENTO NO ANO E PARTICIPACAO DO DETERMINADO SABOR COMO PERCENTUAL

```
-- SABOR, ANO, FATURAMENTO, PARTICIPACAO

SELECT AUX_1.SABOR,
       AUX_1.ANO,
       CONVERT(DECIMAL(15,2),ROUND(AUX_1.FATURAMENTO,2)) 'TOTAL',
       CONVERT(VARCHAR,
               CONVERT(DECIMAL(15,2),
                       ROUND(((AUX_1.FATURAMENTO * 100) / AUX_2.FATURAMENTO),2)
               ) + ' %'
       ) AS 'PERCENTUAL'
FROM (SELECT P.SABOR,
            YEAR(N.DATA) 'ANO',
            SUM(I.QUANTIDADE * I.PREÇO) 'FATURAMENTO'
      FROM [TABELA DE PRODUTOS] P
      INNER JOIN [ITENS NOTAS FISCAIS] I ON I.[CODIGO DO PRODUTO] = P.[CODIGO DO PRODUTO]
      INNER JOIN [NOTAS FISCAIS] N ON N.[NUMERO] = I. [NUMERO]
      WHERE YEAR(N.DATA) = '2016'
      GROUP BY P.SABOR,
               YEAR(N.DATA)
     )
     AUX_1
INNER JOIN (SELECT YEAR(N.DATA) 'ANO',
                  SUM(I.QUANTIDADE * I.PREÇO) 'FATURAMENTO'
            FROM [ITENS NOTAS FISCAIS] I
            INNER JOIN [NOTAS FISCAIS] N ON N.[NUMERO] = I. [NUMERO]
            GROUP BY YEAR(N.DATA)
          ) AUX_2 ON AUX_1.ANO = AUX_2.ANO
```

100 %

Results Messages

	SABOR	ANO	TOTAL	PERCENTUAL
1	Açaí	2016	10024014,1	23.66 %
2	Cereja	2016	1063754,48	2.51 %