ALURA - JOIN

```
SELECT * FROM tabela_de_vendedores;
SELECT * FROM notas_fiscais;
describe tabela_de_vendedores;
show create table tabela_de_vendedores;
describe notas_fiscais;

USE sucos_vendas;

/*Vai mostrar na tela todos os campos da tabela_de_vendedores e todos os campos da tabela notas_fiscais O "A" e um apelido para a tabela "tabela_de_vendedores" e o "B" e um apelido para a tabela de "notas_fiscais" */
SELECT * FROM tabela_de_vendedores A
INNER JOIN notas_fiscais B
ON A.MATRICULA = B.MATRICULA;

/*QUERO SABER QUANTAS NOTAS FISCAIS CADA VENDEDOR EMITIU*/
SELECT A.MATRICULA, A.NOME, COUNT(*) FROM
tabela_de_vendedores A
INNER JOIN notas_fiscais B
ON A.MATRICULA = B.MATRICULA
GROUP BY A.MATRICULA, A.NOME;

/*OUTRA MANEIRA DE FAZER O INNER JOIN, ESSA MANEIRA NÃO E ACONSELHAVEL */
/*QUERO SABER QUANTAS NOTAS FISCAIS CADA VENDEDOR EMITIU*/
SELECT A.MATRICULA, A.NOME, COUNT(*) FROM
tabela_de_vendedores A, notas_fiscais B
WHERE A.MATRICULA = B.MATRICULA
GROUP BY A.MATRICULA, A.NOME;

/*Obtenha o faturamento anual da empresa. Leve em consideração que o valor financeiro das vendas consiste em multiplicar a quantidade pelo preço.*/
SELECT YEAR(DATA_VENDA) AS ANO, SUM(QUANTIDADE * PRECO) AS FATURAMENTO
FROM notas_fiscais NF INNER JOIN itens_notas_fiscais INF
ON NF.NUMERO = INF.NUMERO
GROUP BY YEAR(DATA_VENDA);

DESCRIBE notas_fiscais;

DESCRIBE itens_notas_fiscais;
```

```
/*Verificando quantos clientes eu tenho na minha tabela */
SELECT COUNT(*) FROM tabela_de_clientes;

/*Mostrar todos os clientes que emitiram notas fiscais*/
SELECT cpf, COUNT(*) FROM notas_fiscais GROUP BY cpf;

SELECT * FROM tabela_de_clientes;

SELECT * FROM notas_fiscais;

/*Vai retornar todos os clientes que emitiram notas fiscais*/
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
INNER JOIN notas_fiscais B ON A.CPF = B.CPF;

/*Vai retornar todos os clientes (cpf) da tabela de clientes e só os correspondentes da
tabela de notas fiscais. Conseguir identificar que o cliente "Fábio Carvalho" nunca
comprou na empresa, já que ele nunca emitiu nota fiscal, ou seja,
não tem nenhuma nota fiscal no nome dele */
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
LEFT JOIN notas_fiscais B ON A.CPF = B.CPF;

/*Outra forma de fazer o mesmo select de cima
Retorna os clientes que nunca efetuaram compra na nossa empresa*/
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
LEFT JOIN notas_fiscais B ON A.CPF = B.CPF
WHERE B.CPF IS NULL;/*Vai trazer somente os campos da tabela notas fiscais que são
nulos*/

/*Outra forma de fazer o mesmo select de cima
Retorna os clientes que nunca efetuaram compra na nossa empresa no ano de 2015*/
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
LEFT JOIN notas_fiscais B ON A.CPF = B.CPF
WHERE B.CPF IS NULL AND YEAR(B.DATA_VENDA) = 2015;/*Vai trazer somente os
campos da tabela notas fiscais que são nulos do que forem do ano de 2015*/

/*Trazer todo mundo que e cliente e não tem nota fiscal*/
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM notas_fiscais B
RIGHT JOIN tabela_de_clientes A ON A.CPF = B.CPF
WHERE B.CPF IS NULL;

/*Vai retornar quantos clientes estão cadastrados no total*/
SELECT COUNT(*) FROM tabela_de_clientes;
```

/*O "BAIRRO" seria o campo da tabela tabela_de_clientes e da tabela tabela_de_vendedores
O retorno mostrará os clientes que estão em bairros onde há escritórios da empresa de sucos.
Nessa consulta, obtemos somente 7 registros, o que significa que 8 clientes estão em bairros
que não têm escritório, descobrimos que são 15 clientes cadastrados no total.
Nessa consulta vai retornar todos os campos*/

```
SELECT * FROM tabela_de_vendedores INNER JOIN tabela_de_clientes
ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO;
```

/*Vai retornar somente 2 campos da tabela_de_vendedores e da tabela_de_clientes*/

```
SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.DE_FERIAS,
tabela_de_clientes.BAIRRO,
tabela_de_vendedores.NOME,
tabela_de_clientes.NOME FROM tabela_de_vendedores INNER JOIN tabela_de_clientes
ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO;
```

/*Vai retornar somente 2 campos da tabela_de_vendedores e da tabela_de_clientes*/

```
SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.DE_FERIAS,
tabela_de_clientes.BAIRRO,
tabela_de_vendedores.NOME,
tabela_de_clientes.NOME FROM tabela_de_vendedores LEFT JOIN tabela_de_clientes
ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO;
```

/*Vai retornar todos os clientes e só os vendedores correspondentes*/

```
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME,tabela_de_vendedores.DE_FERIAS,
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;
```

/*Fazendo a consulta utilizando o "CROSS JOIN"*/

```
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME,tabela_de_vendedores.DE_FERIAS,
tabela_de_clientes.BAIRRO,
tabela_de_clientes.NOME FROM tabela_de_vendedores, tabela_de_clientes;
```

/*UNION e o comando que junta duas ou mais tabelas
É importante que as tabelas que serão unidas tenham o mesmo número e tipo de campo O UNION aplica automaticamente o DISTINCT sobre o resultado final da consulta
UNION ALL não aplica o DISTINCT sobre o resultado final da consulta*/

```
SELECT DISTINCT bairro FROM tabela_de_clientes;

SELECT DISTINCT bairro FROM tabela_de_vendedores;
```

/*Vai retornar todos os bairros que os clientes moram e que os vendedores vendem, caso tenha algum bairro que tenha nas duas tabelas, vai aparecer uma unica vez, devido o UNION aplicar automaticamente o DISTINCT sobre o resultado final da consulta*/

```
SELECT DISTINCT bairro FROM tabela_de_clientes
UNION
SELECT DISTINCT bairro FROM tabela_de_vendedores;
```

/*Vai retornar todos os bairros que os clientes moram e que os vendedores vendem, caso tenha algum bairro que tenha nas duas tabelas, vai aparecer esses bairros duplicados, devido o UNION ALL não aplicar automaticamente o DISTINCT sobre o resultado final da consulta*/

```
SELECT DISTINCT bairro FROM tabela_de_clientes
UNION ALL
SELECT DISTINCT bairro FROM tabela_de_vendedores;
```

/*Vai juntar somente os campos iguais da tabela, ou seja, como na tabela de vendedores e de clientes tem esse campo vai juntar e a mesma coisa vai acontecer com o campo "nome"*/

```
SELECT DISTINCT bairro, nome, 'CLIENTE' AS TIPO FROM tabela_de_clientes
UNION
SELECT DISTINCT bairro, nome, 'VENDEDOR' AS TIPO FROM tabela_de_vendedores;
```

/*Vai juntar somente os campos iguais da tabela, ou seja, como na tabela de vendedores e de clientes tem esse campo vai juntar e a mesma coisa vai acontecer com o campo "nome"
AO optar por apelidos diferentes em cada SELECT "TIPO_CLIENTE" e "TIPO_VENDEDOR", apenas o primeiro será considerado. Os nomes das colunas correspondem aos da primeira seleção, ou seja, vai ser considerado somente as colunas/ campos do primeiro select da "tabela_de_clientes"*/

```
SELECT DISTINCT bairro, nome, 'CLIENTE' AS TIPO_CLIENTE FROM tabela_de_clientes
UNION
SELECT DISTINCT bairro, nome, 'VENDEDOR' AS TIPO_VENDEDOR FROM
tabela_de_vendedores;
```

/*SIMULANDO O FULL JOIN com o MYSQL
Enfim, poderemos analisar em uma única consulta quais são os bairros que têm clientes e vendedores (por exemplo, Tijuca); quais possuem vendedores mas não dispõem de compradores cadastrados (Copacabana); e em quais moram clientes porém não têm vendedores (por exemplo, Água Santa).
Essas eram exatamanente as informações que esperávamos do FULL JOIN, ou seja, esse comando pode ser simulado com o LEFT JOIN e o RIGHT JOIN com o UNION entre eles*/

```
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME, DE_FERIAS,
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
UNION
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME, DE_FERIAS,
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;
```

/*Sub-consultas. Podemos usar uma subconsulta dentro de uma consulta*/

```
SELECT DISTINCT BAIRRO FROM tabela_de_vendedores;

SELECT BAIRRO FROM tabela_de_vendedores;
```

/*Quero selecionar todo mundo da tabela de clientes, cujo o bairro seja os bairros que tem escritorio de vendedores*/

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

/*Quero selecionar todo mundo da tabela de clientes, cujo o bairro seja os bairros que tem escritorio de vendedores.
Dessa maneira estou fazendo uma subconsulta
Vou testar somente bairros que estão dentro do resultado do segundo select*/

```
SELECT * FROM tabela_de_clientes WHERE BAIRRO
IN (SELECT DISTINCT BAIRRO FROM tabela_de_vendedores);
```

/*Vou retornar o maior preço dos tipos de embalagens que estão armazenados na tabela de produtos*/

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) FROM tabela_de_produtos
GROUP BY EMBALAGEM;
```

/*Quero saber quais são as embalagens que o maior preco seja maior que 10*/

```
SELECT X.EMBALAGEM, X.PRECO_MAXIMO FROM
(SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS PRECO_MAXIMO FROM
tabela_de_produtos
GROUP BY EMBALAGEM) X WHERE X.PRECO_MAXIMO >= 10;
```

/*Quais foram os clientes que fizeram mais de 2000 compras em 2016?*/
/*Mostre na tela de forma agrupada os campos "cpf" e que mostre somente o total de vendas que sejá acima de 2000 vendas no ano de 2016*/

```
SELECT CPF, COUNT(*) FROM notas_fiscais
WHERE YEAR(DATA_VENDA) = 2016
GROUP BY CPF
HAVING COUNT(*) > 2000;
```

/*Quais foram os clientes que fizeram mais de 2000 compras em 2016?*/
/*Mostre na tela de forma agrupada os campos "cpf" e que mostre somente o total de vendas que sejá acima de 2000 vendas no ano de 2016
Aqui está utilizando subconsultas*/

```
SELECT X.CPF, X.CONTADOR FROM
(SELECT CPF, COUNT(*) AS CONTADOR FROM notas_fiscais
WHERE YEAR(DATA_VENDA) = 2016
GROUP BY CPF) AS X WHERE X.CONTADOR > 2000;

SELECT
        SUB.CPF,
        SUB.QTD
FROM (
        SELECT CPF, COUNT(*) AS QTD
        FROM notas_fiscais
        WHERE YEAR(DATA_VENDA) = 2016
       GROUP BY CPF
       HAVING COUNT(*) > 2000
) AS SUB;
```

/*Vai retornar na tela, quantas vendas os respectivos cpf fizeram */

```
SELECT CPF, COUNT(*) AS contador from notas_fiscais GROUP BY cpf;
```

/*Vai retornar na tela, quantas vendas os respectivos cpf fizeram no ano de 2016*/

```
SELECT CPF, COUNT(*) AS contador from notas_fiscais WHERE YEAR(DATA_VENDA) =
2016 GROUP BY CPF;
```

/*Vai retornar na tela, quantas vendas os respectivos cpf fizeram no ano de 2016 e que as vendas forem maior que 2000*/

```
SELECT CPF, COUNT(*) AS QTD FROM notas_fiscais WHERE YEAR(DATA_VENDA) = 2016
GROUP BY CPF HAVING COUNT(*) > 2000;

/*Quero contar quantidade de embalagens do tipo PET, Garrafa, Lata tem no banco de dados*/
SELECT embalagem, COUNT(*) AS contador FROM tabela_de_produtos GROUP BY
embalagem;

/*View. A View é uma tabela lógica, resultado de uma consulta, que pode ser usada depois em qualquer outra consulta
A View são as minhas visões
Podemos criar uma visão de uma visão no SQL
Podemos criar uma visão de qualquer comando SQL*/

/*Quero saber o maior preco de embalagens do tipo PET, Garrafa, Lata tem no banco de dados*/
SELECT embalagem, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO FROM tabela_de_produtos
GROUP BY embalagem;

/*Quero saber o maior preco de embalagens do tipo PET, Garrafa, Lata tem no banco de dados que sejam maior que 10
Estou utilizando sub-consultas*/
SELECT X.EMBALAGEM, X.MAIOR_PRECO FROM
(SELECT embalagem, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO FROM tabela_de_produtos
GROUP BY embalagem) X WHERE X.MAIOR_PRECO >= 10;

/*Criando uma Visão "View". Que permite eu saber o maior preco de embalagens do tipo PET, Garrafa, Lata tem no banco de dados*/*/
CREATE VIEW `VW_MAIORES_EMBALAGENS` AS
SELECT embalagem, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO FROM tabela_de_produtos
GROUP BY embalagem;

/*Quero saber o maior preco de embalagens do tipo PET, Garrafa, Lata tem no banco de dados que sejam maior que 10
Estou utilizando visão, o "VW_MAIORES_EMBALAGENS" e uma visão que permite eu saber o o maior preco de embalagens do tipo PET, Garrafa, Lata tem no banco de dados */
SELECT X.EMBALAGEM, X.MAIOR_PRECO FROM
VW_MAIORES_EMBALAGENS X WHERE X.MAIOR_PRECO >= 10;

/*Estou fazendo um innerjoin entre uma tabela e uma view, essa view está rodando o group by do maior preco por embalagem
Vai retornar o nome do produto, o tipo de embalagem e o preco de lista do produto, e
```

vai retornar o maior preco por tipo de embalagem*/

```
SELECT A.NOME_DO_PRODUTO, A.EMBALAGEM, A.PRECO_DE_LISTA, X.MAIOR_PRECO
FROM tabela_de_produtos AS A INNER JOIN vw_maiores_embalagens AS X
ON A.EMBALAGEM = X.EMBALAGEM;
```

/*Estou fazendo um innerjoin entre uma tabela e uma view, essa view está rodando o group by do maior preco por embalagem

Vai retornar o nome do produto, o tipo de embalagem e o preco de lista do produto, e vai retornar o maior preco por tipo de embalagem*/

```
SELECT A.NOME_DO_PRODUTO, A.EMBALAGEM, A.PRECO_DE_LISTA, X.MAIOR_PRECO,
((A.PRECO_DE_LISTA / X.MAIOR_PRECO) -1) * 100 AS PERCENTUAL /*Vou ver a variação
do produto mais barato conforme o maior preço dele*/
FROM tabela_de_produtos AS A INNER JOIN vw_maiores_embalagens AS X
ON A.EMBALAGEM = X.EMBALAGEM;
```

/*Estou fazendo um innerjoin entre uma tabela e uma view, essa view está rodando o group by do maior preco por embalagem

Vai retornar o nome do produto, o tipo de embalagem e o preco de lista do produto, e vai retornar o maior preco por tipo de embalagem*/

```
SELECT A.NOME_DO_PRODUTO, A.EMBALAGEM, A.PRECO_DE_LISTA, X.MAIOR_PRECO,
(A.PRECO_DE_LISTA / X.MAIOR_PRECO) * 100 AS PERCENTUAL /*Percentual permite eu
ver quanto ele e mais barato do que o maior preco daquele tipo de embalagem*/
FROM tabela_de_produtos AS A INNER JOIN vw_maiores_embalagens AS X
ON A.EMBALAGEM = X.EMBALAGEM;
```