**Banco de Dados – Consultas SQL - Avançando no SQL com MySQL– Micro-certificação - ALURA**

# Aula 01 – Configurando ambiente e conhecendo o SQL

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Instale o MySQL, conforme o vídeo [**Instalando o MySQL Server**](https://cursos.alura.com.br/course/mysql-manipule-dados-com-sql/task/54324), da aula 1 do curso [**Introdução ao SQL: Manipule dados com MySQL**](https://cursos.alura.com.br/course/mysql-manipule-dados-com-sql).

2) Abra o MYSQL Workbench. Use a conexão LOCALHOST.

3) Clique com o Botão da direita do mouse sobre a lista das bases e escolha Create Schema.

4) Insira o nome Sucos\_Vendas. Clique em Apply duas vezes.

5) Faça Download do arquivo RecuperacaoAmbiente.zip.

6) Descompacte o arquivo.

7) Selecione, na área Navigator, em Administration.

8) Selecione o link Data Import/Restore.

9) Na opção Import From Dump Project Folder escolha o diretório DumpSucosVendas.

10) Selecione Start Import.

11) Verifique na base Sucos\_Vendas se as tabelas foram criadas.

12) Existe outra maneira que importar através de script. Basta executar todos os arquivos .SQL compactados no arquivo RecuperacaoAmbiente.zip.

# O que aprendemos?

Nesta aula, aprendemos:

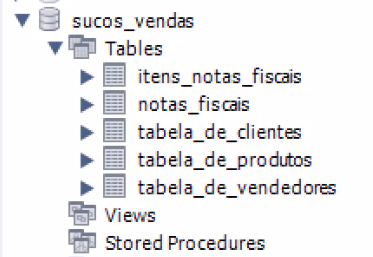
* A recuperar o ambiente;
* Um pouco da história do SQL;
* Um pouco da história do MYSQL;

# Aula 02 - Filtrando as consultas de dados

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Para que possam ser efetuadas as consultas na base de dados, é preciso conhecer as suas tabelas e seus relacionamentos. Para isso, vá no Workbench e verifique se o banco de dados *Sucos\_Vendas* está disponível.

2) Expandindo a árvore de estrutura de base de dados sobre *Sucos\_Vendas*, podemos ver os componentes de um banco de dados. Para as consultas, um dos elementos mais importantes são as tabelas que podem ser vistas em mais detalhe até a sua estrutura de campos.

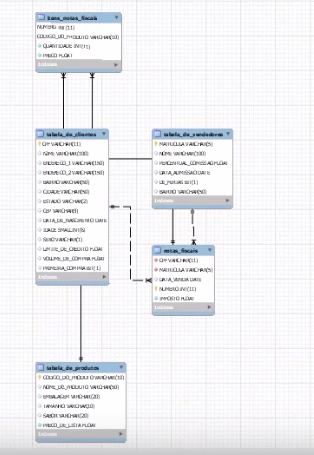


3) Vá no menu e selecione Database / Reverse Engineer.

4) Clique em Next duas vezes e depois escolha o banco no qual a engenharia reversa será efetuada.

5) Continue no assistente confirmando as seleções padrões até o final.

6) Você poderá ver um esquema visual das suas tabelas. Este esquema pode ser um guia para suas consultas.



7) Sabendo como é nossa base, podemos fazer nossas consultas. Selecione um novo script de SQL, com a base de dados selecionada, e digite:

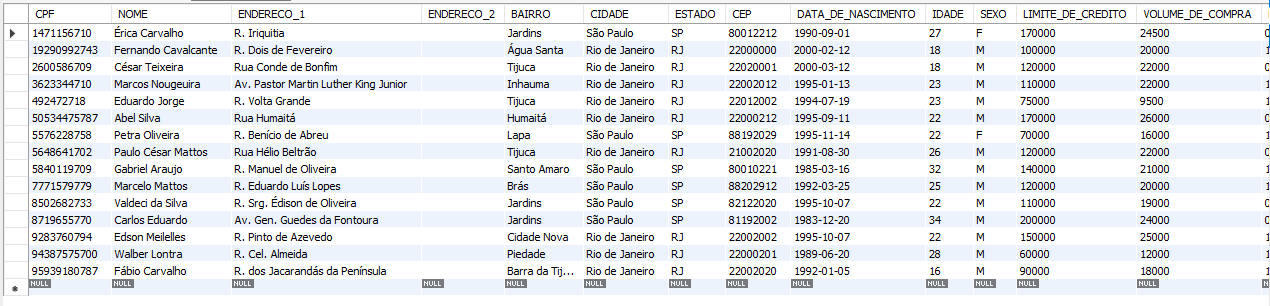
USE sucos\_vendas;

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 tabela\_de\_clientes;



Aqui veremos todos os campos da tabela Tabela\_de\_Clientes. Isso porque os campos foram selecionados um a um.

8) Digite abaixo:

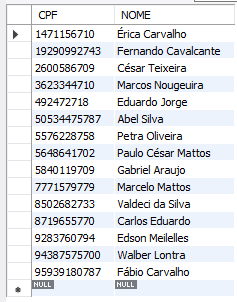
SELECT \* FROM tabela\_de\_clientes;



Este resultado foi semelhante a consulta anterior. Isso porque, ao colocar \* estamos selecionado todos os campos.

9) Digite:

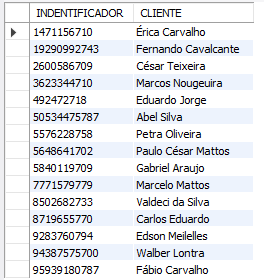
SELECT CPF, NOME FROM tabela\_de\_clientes;



Agora podemos ver que não é necessário selecionar todos os campos de uma tabela. Basta eu destacar os campos que serão vistos.

10) Digite:

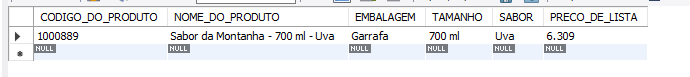
SELECT CPF as INDENTIFICADOR, NOME AS CLIENTE FROM tabela\_de\_clientes;



Nem sempre o nome original da coluna é o nome que queremos que seja retornado pela consulta. Por isso, podemos criar Alias (Apelidos) para os campos escrevendo algo após o comando AS.

11) Podemos filtrar nossa consulta. Digite:

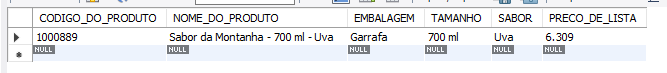
SELECT \* FROM tabela\_de\_produtos WHERE CODIGO\_DO\_PRODUTO = '1000889';



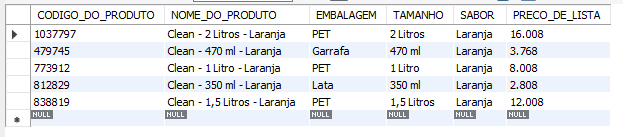
Aqui, o nosso retorno foi uma linha, porque selecionamos um filtro através da chave primária, que não repete.

12) Mas podemos implementar filtros que retornem mais linhas. Veja:

SELECT \* FROM tabela\_de\_produtos WHERE SABOR = 'Uva';



SELECT \* FROM tabela\_de\_produtos WHERE SABOR = 'Laranja';



SELECT \* FROM tabela\_de\_produtos WHERE EMBALAGEM = 'PET';



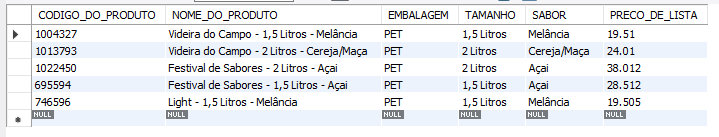
SELECT \* FROM tabela\_de\_produtos WHERE EMBALAGEM = 'pet';



Os filtros usados acima retornam mais linhas. Podemos usar qualquer coluna como critério.

13) Existem comandos de filtro aplicados a valores:

SELECT \* FROM tabela\_de\_produtos WHERE PRECO\_DE\_LISTA > 19.50;



SELECT \* FROM tabela\_de\_produtos WHERE PRECO\_DE\_LISTA BETWEEN 19.50 AND 19.52;

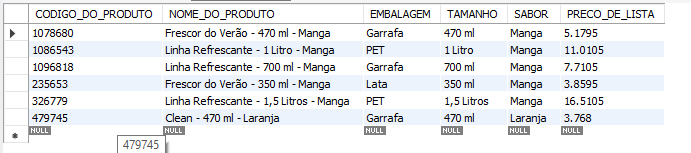


Neste caso podemos usar >,>=, <, <=, =, <> e Between. Assim podemos aplicar filtros sobre os valores que retornem mais valores.

14) É possível aplicar consultas condicionais usando operadores AND e OR. O retorno vai depender do significado do AND e OR numa estrutura lógica. Digite:

SELECT \* FROM tabela\_de\_produtos WHERE SABOR = 'Manga'

OR TAMANHO = '470 ml';

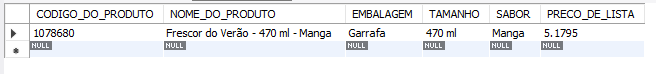


Aqui retornamos ou um filtro (Sabor = Manga) ou outro ( Tamanho = 470 ml). Isso porque usamos o operador OR.

15) Digite:

SELECT \* FROM tabela\_de\_produtos WHERE SABOR = 'Manga'

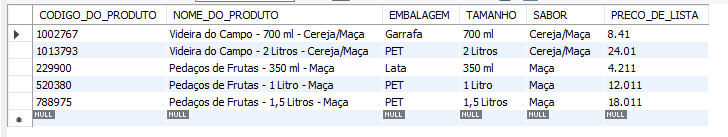
AND TAMANHO = '470 ml';



Agora, por causa do operador AND, o retorno somente ocorrerá quando as duas condições ocorrerem na mesma linha da tabela.

16) Podemos usar parte de um texto para ser usado como critério de localização de registros da tabela. Digite abaixo:

SELECT \* FROM tabela\_de\_produtos WHERE SABOR LIKE '%Maça%';

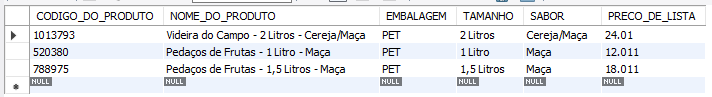


Aqui iremos buscar todos os registros cujo sabor possua a palavra *Maça*. Não importa se no início, no meio ou no final do texto.

17) Podemos mesclar condições LIKE com outras. Digite:

SELECT \* FROM tabela\_de\_produtos WHERE SABOR LIKE '%Maça%'

AND EMBALAGEM = 'PET';



Por fim, faremos a consulta do texto “Maça” apenas para embalagens PET.

# O que aprendemos?

Nesta aula, aprendemos:

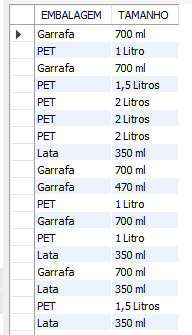
* A importância de conhecer a base de dados antes de fazer as consultas;
* O comando de consultas e como podemos filtrá-las;
* Como podemos mesclar filtros condicionais com AND e OR;
* A usar >, >=, <, <=, = ou <> nos filtros que envolvem valores;
* Como funciona o comando LIKE;

# Aula 03 - Apresentação dos dados de uma consulta

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Voltando ao Workbench, vamos ver formas diferentes de exibir os resultados. Digite:

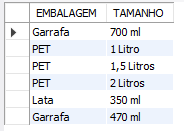
SELECT EMBALAGEM, TAMANHO FROM tabela\_de\_produtos;



Veja que temos linhas onde o conjunto EMBALAGEM / TAMANHO se repete.

2) Agora digite o comando:

SELECT DISTINCT EMBALAGEM, TAMANHO FROM tabela\_de\_produtos;

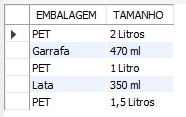


O simples fato de incluirmos a cláusula DISTINCT faz com que os registros não se repitam.

3) Podemos aplicar filtros a seleção com DISTINCT.

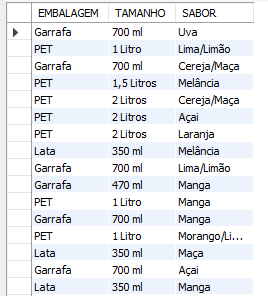
SELECT DISTINCT EMBALAGEM, TAMANHO FROM tabela\_de\_produtos

WHERE SABOR = 'Laranja';



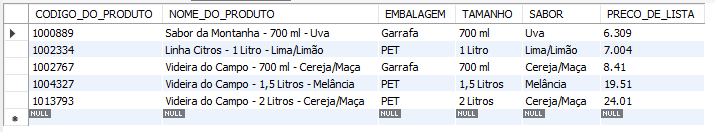
4) E podemos acrescentar mais campos à seleção DISTINCT.

SELECT DISTINCT EMBALAGEM, TAMANHO, SABOR FROM tabela\_de\_produtos;



5) Podemos limitar o número de linhas exibidas na saída. Digite:

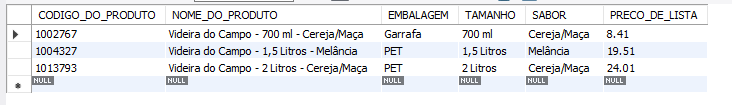
SELECT \* FROM tabela\_de\_produtos LIMIT 5;



Temos acima nossa saída limitada aos primeiros 5 registros.

6) Podemos exibir os registros dentro de um intervalo de linhas. Digite:

SELECT \* FROM tabela\_de\_produtos LIMIT 2,3;



7) As saídas de uma comando SELECT podem ser apresentadas de forma ordenada. Veja abaixo:

SELECT \* FROM tabela\_de\_produtos ORDER BY PRECO\_DE\_LISTA;



Temos os valores ordenados por preço de lista, do menor para o maior.

8) Podemos mudar esta ordem. Digite:

SELECT \* FROM tabela\_de\_produtos ORDER BY PRECO\_DE\_LISTA DESC;



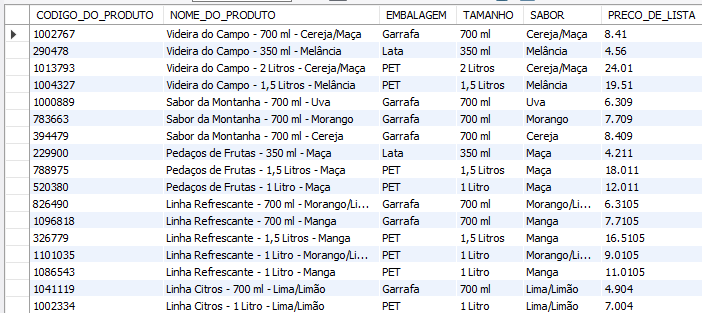
9) Os valores podem vir ordenados alfabeticamente quando incluímos um campo texto no critério de ordenação. Digite:

SELECT \* FROM tabela\_de\_produtos ORDER BY NOME\_DO\_PRODUTO;



10) Também, no critério de ordenação do tipo texto, podemos mudar a ordem para do maior para o menor. Digite:

SELECT \* FROM tabela\_de\_produtos ORDER BY NOME\_DO\_PRODUTO DESC;



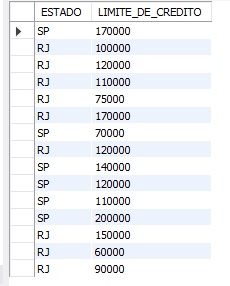
11) O critério de ordenação pode ser diferente para cada tipo. Veja o exemplo abaixo onde usamos dois campos como critério de ordenação e a ordem diferente para cada um deles:

SELECT \* FROM tabela\_de\_produtos ORDER BY EMBALAGEM DESC, NOME\_DO\_PRODUTO ASC;



12) Os dados podem ser agrupados. Quando isso acontece, temos que aplicar um critério de agrupamento para os campos numéricos. Podemos usar SUM, AVG, MAX, MIN, e outros mais. Digite o comando abaixo:

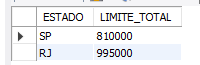
SELECT ESTADO, LIMITE\_DE\_CREDITO FROM tabela\_de\_clientes;



Note que temos várias linhas para RJ e SP. Como fazemos para somar todos os limites de crédito para RJ e SP?

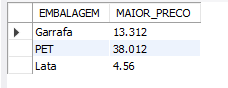
13) A solução está no comando abaixo:

SELECT ESTADO, SUM(LIMITE\_DE\_CREDITO) AS LIMITE\_TOTAL FROM tabela\_de\_clientes GROUP BY ESTADO;



14) Podemos usar outros critérios como o valor máximo.

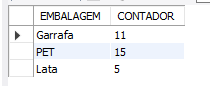
SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS MAIOR\_PRECO FROM tabela\_de\_Produtos GROUP BY EMBALAGEM;



Aqui vemos o maior preço de lista para cada tipo de embalagem.

15) O comando COUNT conta o número de ocorrências na tabela. Digite:

SELECT EMBALAGEM, COUNT(\*) AS CONTADOR FROM tabela\_de\_produtos GROUP BY EMBALAGEM;

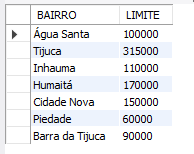


Temos acima o número de produtos PET, Garrafa e Lata.

16) O filtro pode ser aplicado sobre o agrupamento, como uma consulta qualquer. Digite:

SELECT BAIRRO, SUM(LIMITE\_DE\_CREDITO) AS LIMITE FROM tabela\_de\_clientes

WHERE CIDADE = 'Rio de Janeiro' GROUP BY BAIRRO;



17) Além disso, o agrupamento também pode ser feito por mais de um campo. Digite:

SELECT ESTADO, BAIRRO, SUM(LIMITE\_DE\_CREDITO) AS LIMITE FROM tabela\_de\_clientes

GROUP BY ESTADO, BAIRRO;



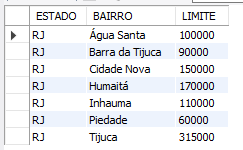
18) Podemos mesclar agrupamento com ordenação. Digite:

SELECT ESTADO, BAIRRO, SUM(LIMITE\_DE\_CREDITO) AS LIMITE FROM tabela\_de\_clientes

WHERE CIDADE = 'Rio de Janeiro'

GROUP BY ESTADO, BAIRRO

ORDER BY BAIRRO;



19) Veja a consulta abaixo:

SELECT ESTADO, SUM(LIMITE\_DE\_CREDITO) AS SOMA\_LIMITE FROM tabela\_de\_clientes

GROUP BY ESTADO;

19.png

20) Queremos aplicar um filtro sobre o resultado desta consulta. Logo digite:

SELECT ESTADO, SUM(LIMITE\_DE\_CREDITO) AS SOMA\_LIMITE FROM tabela\_de\_clientes

WHERE SOMA\_LIMITE > 900000

GROUP BY ESTADO;

Veja que a consulta acima vai ocasionar um erro.

21) Usamos o HAVING para filtrar a saída de uma consulta usando como critério o valor agrupado. Digite:

SELECT ESTADO, SUM(LIMITE\_DE\_CREDITO) AS SOMA\_LIMITE FROM tabela\_de\_clientes

GROUP BY ESTADO HAVING SUM(LIMITE\_DE\_CREDITO) > 900000;

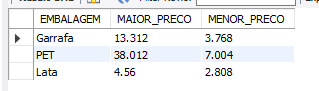
20.png

22) O critério usado no HAVING não precisa ser o mesmo usado no filtro. Veja o comando abaixo:

SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS MAIOR\_PRECO,

MIN(PRECO\_DE\_LISTA) AS MENOR\_PRECO FROM tabela\_de\_produtos

GROUP BY EMBALAGEM;



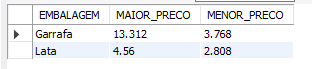
Ele usa o MIN para agrupamento.

23) Porém, na consulta abaixo, o critério do HAVING pede a soma. Digite:

SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS MAIOR\_PRECO,

MIN(PRECO\_DE\_LISTA) AS MENOR\_PRECO FROM tabela\_de\_produtos

GROUP BY EMBALAGEM HAVING SUM(PRECO\_DE\_LISTA) <= 80;



24) No HAVING podemos usar mais de um critério usando AND ou OR.

SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS MAIOR\_PRECO,

MIN(PRECO\_DE\_LISTA) AS MENOR\_PRECO FROM tabela\_de\_produtos

GROUP BY EMBALAGEM HAVING SUM(PRECO\_DE\_LISTA) <= 80 AND MAX(PRECO\_DE\_LISTA) >= 5;

23.png

25) O comando CASE permite que possa ser classificado cada registro da tabela. Digite o comando abaixo:

SELECT NOME\_DO\_PRODUTO, PRECO\_DE\_LISTA,

CASE

WHEN PRECO\_DE\_LISTA >= 12 THEN 'PRODUTO CARO'

WHEN PRECO\_DE\_LISTA >= 7 AND PRECO\_DE\_LISTA < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO'

END AS STATUS\_PRECO

FROM tabela\_de\_produtos;



Com o CASE foi possível classificar os produtos como CARO, BARATO ou EM CONTA conforme o valor do seu preço de lista.

26) Podemos usar o CASE como critério de agrupamento, Digite o comando abaixo:

SELECT EMBALAGEM,

CASE

WHEN PRECO\_DE\_LISTA >= 12 THEN 'PRODUTO CARO'

WHEN PRECO\_DE\_LISTA >= 7 AND PRECO\_DE\_LISTA < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO'

END AS STATUS\_PRECO, AVG(PRECO\_DE\_LISTA) AS PRECO\_MEDIO

FROM tabela\_de\_produtos

WHERE sabor = 'Manga'

GROUP BY EMBALAGEM,

CASE

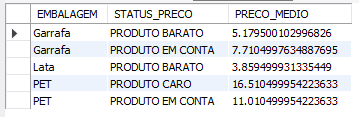
WHEN PRECO\_DE\_LISTA >= 12 THEN 'PRODUTO CARO'

WHEN PRECO\_DE\_LISTA >= 7 AND PRECO\_DE\_LISTA < 12 THEN 'PRODUTO EM CONTA'

ELSE 'PRODUTO BARATO'

END

ORDER BY EMBALAGEM;



# O que aprendemos?

Nesta aula, aprendemos:

* Como apresentar somente linhas distintas numa seleção;
* Como ordenar a saída da seleção;
* Como agrupar linhas por um conjunto de campos e aplicando um critério de agrupamento sobre os campos numéricos (SOMA, MIN, MAX, MÉDIA, etc ..).
* Como utilizar o comando HAVING para aplicar um filtro usando os campos numéricos agrupados como condição.
* Como limitar a saída das consultas;
* Como usar o CASE para classificar um determinado campo por um critério.

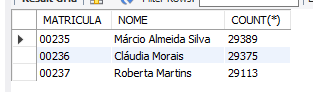
# Aula 04 - Juntando tabelas e consultas

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

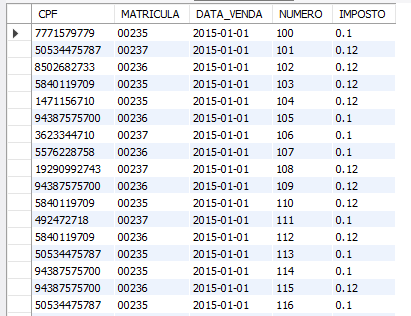
1) Aqui veremos como conectar as consultas de tabelas diferentes. Chamamos esta união de JOIN.

2) Veja o conteúdo de duas tabelas digitando os comandos abaixo:

SELECT \* FROM tabela\_de\_vendedores;



SELECT \* FROM notas\_fiscais;



3) Podemos conectar essas duas tabelas pelo campo em comum (MATRICULA). Digite:

SELECT \* FROM tabela\_de\_vendedores A

INNER JOIN notas\_fiscais B

ON A.MATRICULA = B.MATRICULA;



4) Podemos aplicar agrupamentos ao resultado da consulta que conecta uma ou mais tabelas:

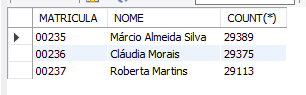
SELECT A.MATRICULA, A.NOME, COUNT(\*) FROM

tabela\_de\_vendedores A

INNER JOIN notas\_fiscais B

ON A.MATRICULA = B.MATRICULA

GROUP BY A.MATRICULA, A.NOME;



5) Nem sempre todas as linhas podem ser conectadas. Existem outros tipos de JOINs que nos permite identificar quem não pode ser conectado. Veja a consulta abaixo:

SELECT COUNT(\*) FROM tabela\_de\_clientes;

5.png

Ela mostra que temos 15 clientes.

6) Vamos fazer um JOIN com a tabela de notas fiscais e ver quantos clientes possuem notas emitidas. Digite:

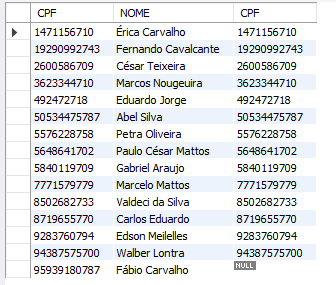


Se você contar verá que, na consulta acima, temos 14 linhas. Existe um cliente que está no cadastro mas não teve nota fiscal emitida.

7) Podemos usar o LEFT JOIN. Digite:

SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela\_de\_clientes A

LEFT JOIN notas\_fiscais B ON A.CPF = B.CPF



O cliente que possui o CPF vindo da tabela de notas com o valor nulo, é o cliente que nunca emitiu nota fiscal.

8) A seleção correta seria:

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;

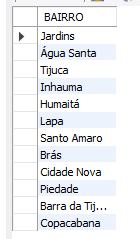
8.png

9) Podemos juntar duas ou mais consultas, Desde que os campos selecionados sejam os mesmos. Digite:

SELECT DISTINCT BAIRRO FROM tabela\_de\_clientes

UNION

SELECT DISTINCT BAIRRO FROM tabela\_de\_vendedores;

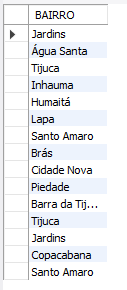


10) O UNION ALL não faz a seleção com um DISTINCT. As linhas se repetem se existirem em ambas as tabelas. Digite:

SELECT DISTINCT BAIRRO FROM tabela\_de\_clientes

UNION ALL

SELECT DISTINCT BAIRRO FROM tabela\_de\_vendedores;



Veja que Santo Amaro aparece duas vezes. Uma vindo da tabela de clientes e outra da tabela de produtos.

11) Podemos simular o FULL JOIN, que não é suportado pelo MYSQL, usando o LEFT JOIN e RIGHT JOIN com UNION. Digite:

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

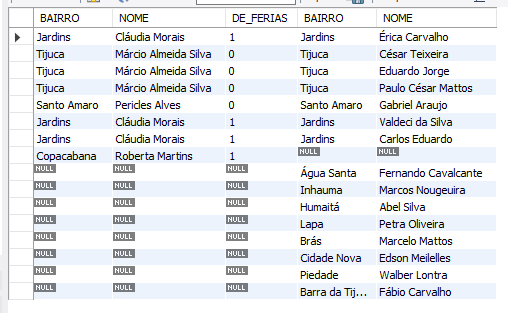
SELECT tabela\_de\_vendedores.BAIRRO,

tabela\_de\_vendedores.NOME, DE\_FERIAS,

tabela\_de\_clientes.BAIRRO,

tabela\_de\_clientes.NOME FROM tabela\_de\_vendedores RIGHT JOIN tabela\_de\_clientes

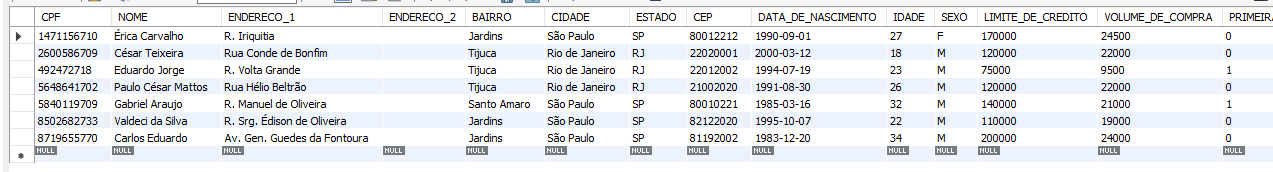
ON tabela\_de\_vendedores.BAIRRO = tabela\_de\_clientes.BAIRRO;



12) As sub-consultas permitem que possa ser feita seleções usando como critérios outras seleções. Digite:

SELECT \* FROM tabela\_de\_clientes WHERE BAIRRO

IN (SELECT DISTINCT BAIRRO FROM tabela\_de\_vendedores);

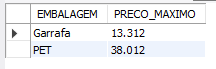


13) Podemos aplicar uma consulta, em vez de sobre uma tabela, sobre outra consulta. Digite:

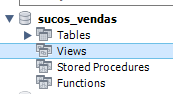
SELECT X.EMBALAGEM, X.PRECO\_MAXIMO FROM

(SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS PRECO\_MAXIMO FROM tabela\_de\_produtos

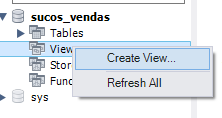
GROUP BY EMBALAGEM) X WHERE X.PRECO\_MAXIMO >= 10;



14) Podemos transformar uma consulta numa visão (View) que depois pode ser usada em outras consultas como uma tabela. Crie a visão. Para isso, expanda na árvore do canto esquerdo, onde temos o nome do banco, e vá em Views.



15) Botão da direita do mouse sobre Views e crie uma nova visão.



16) Digite o seguinte comando:

CREATE VIEW `VW\_MAIORES\_EMBALAGENS` AS

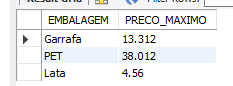
SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS PRECO\_MAXIMO FROM tabela\_de\_produtos

GROUP BY EMBALAGEM

17) Clique em Apply e siga os passos até a criação da visão.

18) Podemos manipular a visão como uma tabela. Digite:

SELECT \* FROM VW\_MAIORES\_EMBALAGENS;

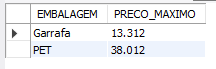


19) Logo a consulta:

SELECT X.EMBALAGEM, X.PRECO\_MAXIMO FROM

(SELECT EMBALAGEM, MAX(PRECO\_DE\_LISTA) AS PRECO\_MAXIMO FROM tabela\_de\_produtos

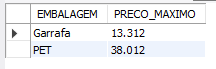
GROUP BY EMBALAGEM) X WHERE X.PRECO\_MAXIMO >= 10;



Pode ser substituída por:

SELECT X.EMBALAGEM, X.PRECO\_MAXIMO FROM

VW\_MAIORES\_EMBALAGENS X WHERE X.PRECO\_MAXIMO >= 10;



# O que aprendemos?

Nesta aula, aprendemos:

* Como conectar duas ou mais tabelas através de JOINs;
* Os tipos de JOINs existentes e quais são suportados pelo MYSQL;
* Os comandos UNION e UNION ALL, para juntar duas ou mais seleções desde que tenham os mesmos campos selecionados;
* Usar uma consulta como critério de filtro de outra consulta;
* Como usar uma consulta dentro de outra consulta;
* Criar e usar visões (Views);

# Aula 05 - Funções do MYSQL

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

1) Nesta aula iremos ver exemplos de funções.

2) Primeiro vimos as funções do tipo texto. Veja alguns exemplos com seus respectivos retornos:

SELECT LTRIM(' OLÁ') AS RESULTADO;

1.png

SELECT RTRIM('OLÁ ') AS RESULTADO;

2.png

SELECT TRIM(' OLÁ ') AS RESULTADO;

3.png

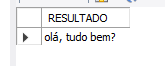
SELECT CONCAT('OLÁ', ' ', 'TUDO BEM','?') AS RESULTADO;

4.png

SELECT UPPER('olá, tudo bem?') AS RESULTADO;

5.png

SELECT LOWER('OLÁ, TUDO BEM?') AS RESULTADO;



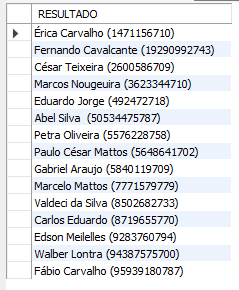
SELECT SUBSTRING('OLÁ, TUDO BEM?', 6) AS RESULTADO;

7.png

SELECT SUBSTRING('OLÁ, TUDO BEM?', 6, 4) AS RESULTADO;

8.png

SELECT CONCAT(NOME, ' (', CPF, ') ') AS RESULTADO FROM TABELA\_DE\_CLIENTES;



3) Temos as funções de datas. Execute os comandos abaixo:

SELECT CURDATE();

10.png

SELECT CURRENT\_TIME();

11.png

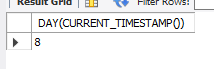
SELECT CURRENT\_TIMESTAMP();

12.png

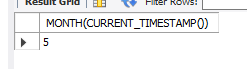
SELECT YEAR(CURRENT\_TIMESTAMP());

13.png

SELECT DAY(CURRENT\_TIMESTAMP());



SELECT MONTH(CURRENT\_TIMESTAMP());



SELECT MONTHNAME(CURRENT\_TIMESTAMP());

16.png

SELECT DATEDIFF(CURRENT\_TIMESTAMP(), '2019-01-01') AS RESULTADO;

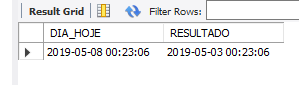
17.png

SELECT DATEDIFF(CURRENT\_TIMESTAMP(), '1965-09-04') AS RESULTADO;

18.png

SELECT CURRENT\_TIMESTAMP() AS DIA\_HOJE

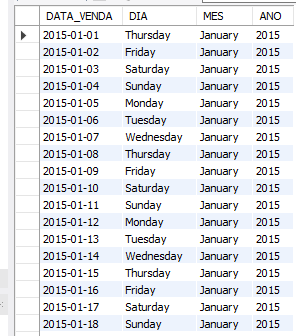
, DATE\_SUB(CURRENT\_TIMESTAMP(), INTERVAL 5 DAY) AS RESULTADO;



SELECT DISTINCT DATA\_VENDA,

DAYNAME(DATA\_VENDA) AS DIA, MONTHNAME(DATA\_VENDA) AS MES

, YEAR(DATA\_VENDA) AS ANO FROM NOTAS\_FISCAIS;



4) Alguns exemplos de funções matemáticas:

SELECT (23+((25-2)/2)\*45) AS RESULTADO;

21.png

SELECT CEILING(12.33333232323) AS RESULTADO;

22.png

SELECT ROUND(12.7777232323) AS RESULTADO;

23.png

SELECT FLOOR(12.7777232323) AS RESULTADO;

24.png

SELECT RAND() AS RESULTADO;

25.png

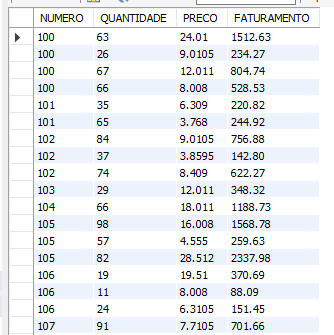
SELECT NUMERO, QUANTIDADE, PRECO, QUANTIDADE \* PRECO AS FATURAMENTO

FROM ITENS\_NOTAS\_FISCAIS;



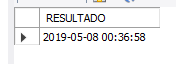
SELECT NUMERO, QUANTIDADE, PRECO, ROUND(QUANTIDADE \* PRECO, 2) AS FATURAMENTO

FROM ITENS\_NOTAS\_FISCAIS;



5) Temos também funções de conversão. Execute os exemplos abaixo:

SELECT CURRENT\_TIMESTAMP() AS RESULTADO;



SELECT CONCAT('O dia de hoje é : ', CURRENT\_TIMESTAMP()) AS RESULTADO;

29.png

SELECT CONCAT('O dia de hoje é : ',

DATE\_FORMAT(CURRENT\_TIMESTAMP(),'%W, %d/%m/%Y - %U') ) AS RESULTADO;

30.png

SELECT SUBSTRING(CONVERT(23.3, CHAR),1,1) AS RESULTADO;

31.png

# O que aprendemos?

Nesta aula, aprendemos:

* Algumas funções de texto;
* Funções matemáticas;
* Foi mostrado funções de datas;
* Abordamos funções de conversão.

# Aula 06 - Exemplos de relatórios

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com os próximos cursos que tenham este como pré-requisito.

1) Vamos por em prática o nosso conhecimento.

2) Primeiro montamos uma seleção que determina se as vendas mensais por cliente são válidas ou não. Consideramos válidas vendas abaixo da quantidade limite e não válidas acima da quantidade limite existente no cadastro do cliente. A consulta é a mostrada abaixo:

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

CASE WHEN (X.QUANTIDADE\_LIMITE - X.QUANTIDADE\_VENDAS) < 0 THEN 'INVÁLIDA'

ELSE 'VÁLIDA' END AS STATUS\_VENDA

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;



3) Outro exemplo de relatório é o que determina a venda por sabores, para o ano de 2016, apresentando o percentual de participação de cada um destes sabores, ordenados.

SELECT VENDA\_SABOR.SABOR, VENDA\_SABOR.ANO, VENDA\_SABOR.QUANTIDADE,

ROUND((VENDA\_SABOR.QUANTIDADE/VENDA\_TOTAL.QUANTIDADE) \* 100, 2) AS PARTICIPACAO FROM

(SELECT TP.SABOR, YEAR(NF.DATA\_VENDA) AS ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE FROM

TABELA\_DE\_PRODUTOS TP

INNER JOIN ITENS\_NOTAS\_FISCAIS INF ON TP.CODIGO\_DO\_PRODUTO = INF.CODIGO\_DO\_PRODUTO

INNER JOIN NOTAS\_FISCAIS NF ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA\_VENDA) = 2016

GROUP BY TP.SABOR, YEAR(NF.DATA\_VENDA)) AS VENDA\_SABOR

INNER JOIN

(SELECT YEAR(NF.DATA\_VENDA) AS ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE FROM

TABELA\_DE\_PRODUTOS TP

INNER JOIN ITENS\_NOTAS\_FISCAIS INF ON TP.CODIGO\_DO\_PRODUTO = INF.CODIGO\_DO\_PRODUTO

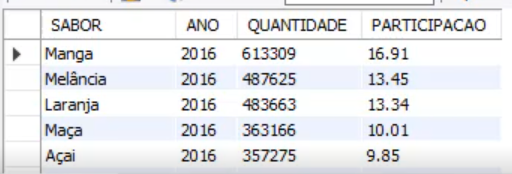
INNER JOIN NOTAS\_FISCAIS NF ON NF.NUMERO = INF.NUMERO

WHERE YEAR(NF.DATA\_VENDA) = 2016

GROUP BY YEAR(NF.DATA\_VENDA)) AS VENDA\_TOTAL

ON VENDA\_SABOR.ANO = VENDA\_TOTAL.ANO

ORDER BY VENDA\_SABOR.QUANTIDADE DESC



# O que aprendemos?

Nesta aula, aprendemos:

* Colocarmos em prática o nosso conhecimento montando dois relatórios conforme especificado pela empresa de suco de frutas.