

Consultas avançadas com SQL, no MySQL

Este documento será um documento mais objetivo, pois explicações mais básicas estão contidas no documento introdução ao SQL no MySQL.

Iniciando nas pesquisas com selects mais completos e dinâmicos

Começando por uma consulta utilizando o **BETWEEN**, realizar uma pesquisa para buscas parâmetros entre dois itens, mais usado em casos onde o campo é do tipo float.

```
SELECT * FROM tabela_de_produtos WHERE PRECO_DE_LISTA BETWEEN 19.50  
AND 19.52;
```

DISTINCT > Utilizado para trazer uma pesquisa mais objetiva, um exemplo seria uma tabela de clientes com uma coluna de cidades, colocando o **distinct** ele irá trazer todas as cidades sem repetições da mesma, dessa forma temos como saber quais são as cidades cadastradas na tabela de uma forma mais fácil e ágil.

```
SELECT DISTINCT EMBALAGEM, TAMANHO FROM tabela_de_produtos;
```

ORDER BY > Utilizado para ordenar os resultados das pesquisas, essa ordenação pode ser tanto numérica quanto alfabética, temos duas formas de ordenação a primeira seria **asc** (ascendente) do menor para o maior e **desc** (descendente) do maior para o menor, no caso de duas ordenações, será efetuado a ordenação da primeira coluna e logo em seguida da segunda coluna, seguindo a ordenação da primeira.

```
SELECT * FROM tabela_de_produtos ORDER BY PRECO_DE_LISTA;
```

```
SELECT * FROM tabela_de_produtos ORDER BY PRECO_DE_LISTA DESC;
```

```
SELECT * FROM tabela_de_produtos ORDER BY EMBALAGEM DESC, NOME_DO_PRODUTO ASC;
```

Os dados podem ser agrupados. Quando isso acontece, temos que aplicar um critério de agrupamento para os campos numéricos. Podemos usar **SUM**(para soma), **AVG**(para calcular a média dos campos), **MAX**(para pegar o valor mais alto da coluna ou tabela), **MIN**(para pegar o menor valor de um campo ou tabela) e **COUNT**(para trazer quantos itens existentes naquela coluna ou tabela).

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS LIMITE_TOTAL FROM tabela_de_clientes GROUP BY ESTADO;
```

OBS: O AS é utilizado para apelidar um objeto dentro do SQL então a partir do momento em que foi colocado `AS LIMITE_TOTAL` essa coluna começou a se chamar `LIMITE_TOTAL` como um apelido.

GRUPY BY > Usado para agrupar campos, vamos pegar como exemplo a soma de um campo, se executamos o comando abaixo sem o **GRUPY BY** ele retornará uma pesquisa com o limite total da tabela, mas aplicando o **GRUPY BY ESTADO** ele irá retornar o valor total de cada estado, com essa explicação temos que o comando **GRUPY BY** é utilizado para diferenciar e agrupar itens dentro do SQL.

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS LIMITE_TOTAL FROM tabela_de_clientes GROUP BY ESTADO;
```

Nesse comando também podemos adicionar condições, podemos agrupar por mais de um campo e utilizar ainda o comando **ORDER BY** como nos exemplos abaixo:

```
SELECT BAIRRO, SUM(LIMITE_DE_CREDITO) AS LIMITE FROM tabela_de_clientes  
WHERE CIDADE = 'Rio de Janeiro' GROUP BY BAIRRO;
```

```
SELECT ESTADO, BAIRRO, SUM(LIMITE_DE_CREDITO) AS LIMITE FROM tabela_de_clientes  
GROUP BY ESTADO, BAIRRO;
```

```
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;
```

HAVING > Usamos o **HAVING** para filtrar a saída de uma consulta usando como critério o valor agrupado.

```
SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS SOMA_LIMITE FROM tabela_de_clientes  
GROUP BY ESTADO HAVING SUM(LIMITE_DE_CREDITO) > 900000;
```

```
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO,  
MIN(PRECO_DE_LISTA) AS MENOR_PRECO FROM tabela_de_produtos  
GROUP BY EMBALAGEM;
```

```
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;
```

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;
```

CASE > Utilizado para efetuar pesquisas passando condições e testes lógicos como parâmetro, vamos aderir o seguinte caso, onde temos alunos com uma variedade de nota, porém precisamos identificar quais foram os alunos que passaram e que reprovaram, para isso podemos montar uma pesquisa utilizando o CASE, passando como teste lógico os parâmetros que os alunos que ficam acima de 7 passaram de ano e os que ficaram abaixo de 7 reprovaram onde teremos no retorno uma nova coluna informando essa ocasião.

```
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;
```

Podemos também utilizar o GROUP BY para realizar os agrupamentos dos itens:

```
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
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;
```

Podemos adicionar um filtro nesse código:

```
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;
```

INNER JOIN → Utilizado para realizar uma junção de tabelas onde só é retornado os itens relacionados entre si, deixando de mostrar os itens das tabelas que não possuem relacionamentos.

```
SELECT * FROM tabela_de_vendedores A
INNER JOIN notas_fiscais B
ON A.MATRICULA = B.MATRICULA;
```

Aplicando um agrupamento no inner join:

```
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;
```

LEFT JOIN → Utilizado para trazer todos os itens da tabela da esquerda e somente os relacionados da tabela da direita.

```
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
LEFT JOIN notas_fiscais B ON A.CPF = B.CPF
```

	CPF	NOME	CPF
▶	1471156710	Érica Carvalho	1471156710
	19290992743	Fernando Cavalcante	19290992743
	2600586709	César Teixeira	2600586709
	3623344710	Marcos Nogueira	3623344710
	492472718	Eduardo Jorge	492472718
	50534475787	Abel Silva	50534475787
	5576228758	Petra Oliveira	5576228758
	5648641702	Paulo César Mattos	5648641702
	5840119709	Gabriel Araujo	5840119709
	7771579779	Marcelo Mattos	7771579779
	8502682733	Valdeci da Silva	8502682733
	8719655770	Carlos Eduardo	8719655770
	9283760794	Edson Meilletes	9283760794
	94387575700	Walber Lontra	94387575700
	95939180787	Fábio Carvalho	NULL

```
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;
```

	CPF	NOME	CPF
▶	95939180787	Fábio Carvalho	NULL

RIGTH JOIN → Esse comando traz todos os itens da direita e apenas os relacionados a esquerda, a estrutura e a mesma dos JOINS à cima.

```
SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A
RIGTH JOIN notas_fiscais B ON A.CPF = B.CPF
WHERE B.CPF IS NULL;
```

UNION → Podemos juntar duas consultas ou mais utilizando o comando UNION:

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

FULL JOIN → Esse comando não é suportado pela ferramenta Mysql, mas podemos simular ele da seguinte forma:

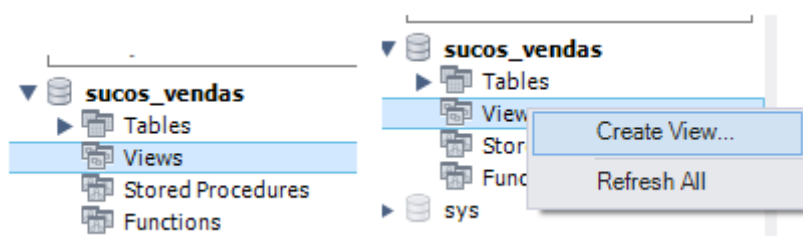
```
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME, DE_FERIAS,
tabela_de_clientes.BAIRRO,
tabela_de_clientes.NOME FROM tabela_de_vendedores LEFT JOIN
tabela_de_clientes
ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO
UNION
SELECT tabela_de_vendedores.BAIRRO,
tabela_de_vendedores.NOME,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

Utilizando sub-consultas

As sub-consultas nos permitem que possamos realizar seleções usando como critérios outras seleções.

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

Podemos criar também visões de nosso banco que são nada mais nada menos que pesquisas que se transformam em tabelas, são uteis para pesquisas rápidas e também para casos onde pessoas necessitam visualizar nosso banco, dessa forma podemos dar a elas somente o necessário para o determinado trabalho.



```
CREATE VIEW `VW_MAIORES_EMBALAGENS` AS
SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS PRECO_MAXIMO FROM
tabela_de_produtos
GROUP BY EMBALAGEM
```

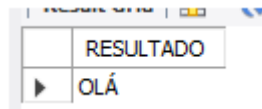
Ao criar uma VIEW temos que seguir um padrão de nomenclatura que seria colocar VW na frente do nome dessa tabela, para identificar que essa tabela esta é uma tabela visão.

Funções:

Texto:

Ltrim → Remover os espaços à direita do texto

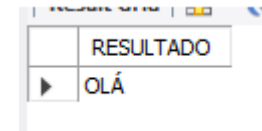
```
SELECT LTRIM(' OLÁ') AS RESULTADO;
```



RESULTADO
OLÁ

Rtrim → Remover os espaços a esquerda da strig.

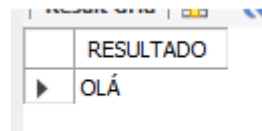
```
SELECT RTRIM('OLÁ ') AS RESULTADO;
```



RESULTADO
OLÁ

Trim → Remover os espaços da esquerda e direita.

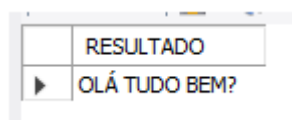
```
SELECT TRIM(' OLÁ ') AS RESULTADO;
```



RESULTADO
OLÁ

Concat → Fazer a união de duas strings.

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



RESULTADO
OLÁ TUDO BEM?

Upper → Deixar todas as letras da string em maiúsculo.

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

	RESULTADO
▶	OLÁ, TUDO BEM?

Lower → Deixa todas as letras em minúsculo.

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

	RESULTADO
▶	olá, tudo bem?

Substring → Basicamente utilizando essa função você pode selecionar qual posição da string deseja que seja exibida na tela.

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

	RESULTADO
▶	TUDO

Data:

Curdate() → Pegar a data atual da máquina.

	CURDATE()
▶	2019-05-08

Current_time() → Pegar horário da máquina ou de algum item do banco com tipo date time.

```
SELECT CURRENT_TIME ();
```

	CURRENT_TIME()
▶	00:23:06

Current_timestamp() → Pegar data e hora.

```
SELECT CURRENT_TIMESTAMP ();
```

	CURRENT_TIMESTAMP()
▶	2019-05-08 00:23:06

Year → Pegar somente o ano.

```
SELECT YEAR (CURRENT_TIMESTAMP ());
```

	YEAR(CURRENT_TIMESTAMP())
▶	2019

DAY → Selecionar somente o dia.

```
SELECT DAY (CURRENT_TIMESTAMP ());
```

	MONTH(CURRENT_TIMESTAMP())
▶	5

Monthname → Seleciona o nome do mês.

```
SELECT MONTHNAME (CURRENT_TIMESTAMP ());
```

	MONTHNAME(CURRENT_TIMESTAMP())
▶	May

```
SELECT DATEDIFF (CURRENT_TIMESTAMP (), '2019-01-01') AS RESULTADO;
```

```
SELECT DATEDIFF (CURRENT_TIMESTAMP (), '1965-09-04') AS RESULTADO;
```

```
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;
```

Matemáticos:

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

```
SELECT CEILING (12.33333232323) AS RESULTADO;
```

```
SELECT ROUND (12.7777232323) AS RESULTADO;
```

```
SELECT FLOOR (12.7777232323) AS RESULTADO;
```

```
SELECT RAND () AS RESULTADO;
```

```
SELECT NUMERO, QUANTIDADE, PRECO, QUANTIDADE * PRECO AS FATURAMENTO  
FROM ITENS_NOTAS_FISCAIS;
```

```
SELECT CURRENT_TIMESTAMP () AS RESULTADO;
```

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



```

• SELECT VENDA_TAMANHO.TAMANHO, VENDA_TAMANHO.ANO, VENDA_TAMANHO.QUANTIDADE,
  ROUND((VENDA_TAMANHO.QUANTIDADE/VENDA_TOTAL.QUANTIDADE) * 100, 2) AS PARTICIPACAO FROM
  (SELECT TP.TAMANHO, 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.TAMANHO, YEAR(NF.DATA_VENDA)) AS VENDA_TAMANHO
  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_TAMANHO.ANO = VENDA_TOTAL.ANO
  ORDER BY VENDA_TAMANHO.QUANTIDADE DESC

```

```

• 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, (1 - (X.QUANTIDADE_LIMITE/X.QUANTIDADE_VENDAS)) * 100 AS PERCENTUAL
  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'))
  WHERE (X.QUANTIDADE_LIMITE - X.QUANTIDADE_VENDAS) < 0;

```