Banco de Dados

Aula 11 - Linguagem SQL – Consulta avançada de tabelas

Apresentação

Na aula anterior, vimos os comandos SHOW DATABASES, SHOW TABLES e DESC que são utilizados para visualização dos dados e estruturas de bancos de dados e tabelas associadas a eles. Em seguida, foi iniciado o estudo do comando SELECT, que é o principal mecanismo de consultas da linguagem SQL. Este comando trabalha em conjunto com as cláusulas FROM e WHERE, a primeira define as tabelas que serão usadas na consulta e a segunda quais condições são usadas para filtrar os resultados.

Analisamos em detalhes diferentes modos de construção da cláusula WHERE usando operadores de comparação (<, <=, >, >=, <>, =), operadores lógicos (AND e OR) e cláusulas complementares (BETWEEN, LIKE, IN e IS NULL). Ao final da aula, a partir dessas informações já estávamos aptos a realizar consultas simples em nossas tabelas.

Agora, podemos refinar as nossas consultas facilitando a compreensão dos resultados, ordenando os dados, extraindo novas informações dos dados existentes ou melhorando a apresentação dos mesmos. Nesta aula, vamos aprender a ordenar e a agrupar os dados fornecidos em uma consulta e conheceremos funções especiais e cláusulas auxiliares que nos ajudarão a apresentar nossos resultados de forma adequada.

Vídeo 01 - Apresentação da Aula

Objetivos

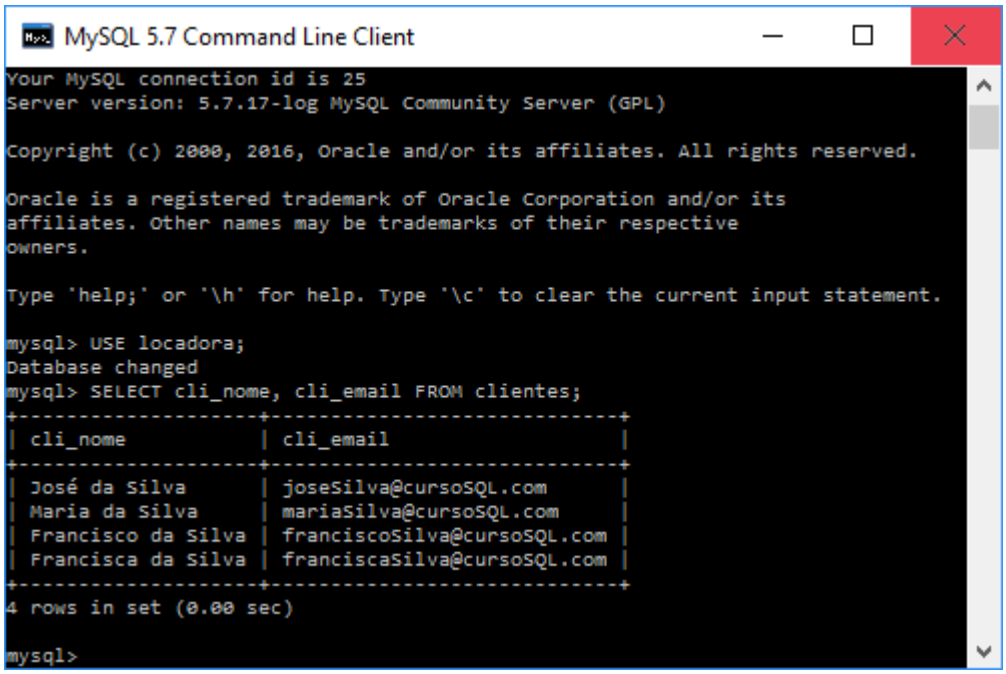
- Consultar dados ordenados em tabelas.
- Consultar dados agrupados em tabelas.
- Utilizar funções especiais para manipulação e apresentação das pesquisas.

Consultas usando ordenação de dados

Na aula anterior, aprendemos a fazer consultas simples aos dados da tabela **clientes** e **filmes** do nosso banco de dados **locadora**. Por exemplo, para obter uma lista dos nomes dos clientes com seus respectivos e-mails, utilizamos o comando SELECT, conforme apresentado no quadro abaixo, obtém-se o resultado ilustrado na **Figura 1**.

```
mysql> SELECT  cli_nome, cli_email
   FROM clientes;
```

Figura 01 - Tela do MySQL após o comando SELECT cli_clientes, cli_email FROM **clientes**.



Fonte: MySQL 5.7 *Command Line Client*

Observe que a coluna com o nome dos clientes não está em ordem alfabética. Na verdade, o resultado é apresentado na ordem em que os dados são inseridos no banco de dados. Nesse caso, não temos dificuldades de achar o e-mail da cliente Maria da Silva, por exemplo, pois temos apenas 4 linhas inseridas no banco de dados. Continuaria sendo fácil se tivéssemos 100, 200 ou 300 registros? Claro que

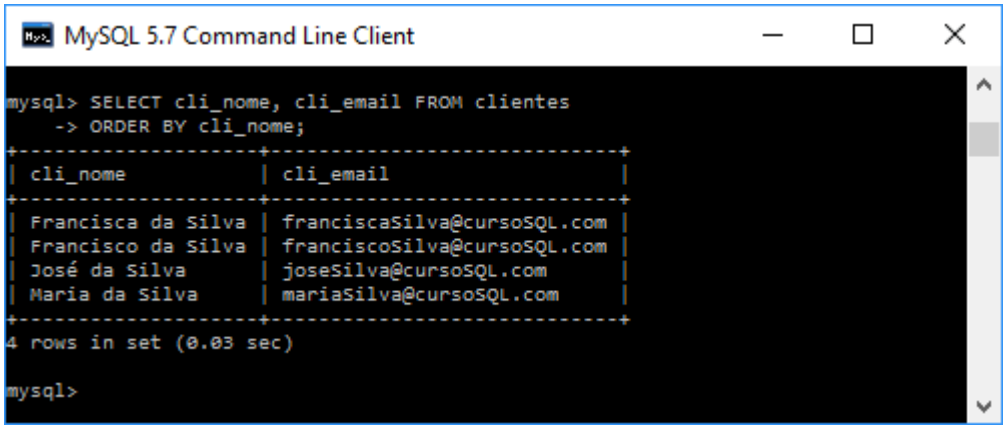
não, perderíamos muito tempo fazendo isso. Para facilitar nossa vida, o comando SELECT possui a cláusula **ORDER BY** que ordena os dados de uma consulta, facilitando a visualização dos resultados.

Continuando com o mesmo exemplo, para obter uma lista dos nomes dos clientes com seus respectivos e-mails, apresentando o resultado considerando o nome dos clientes em ordem alfabética, utilizamos o comando SELECT, conforme apresentado no quadro a seguir.

```
mysql> SELECT  cli_nome, cli_email
   FROM clientes
   ORDER BY cli_nome;
```

Observe que agora as informações sobre os nomes e e-mails são apresentadas em ordem alfabética de acordo com o nome do cliente, conforme é ilustrado na **Figura 2**.

Figura 02 - Tela do MySQL após o comando **SELECT cli_clientes, cli_email FROM clientes ORDER BY cli_nome**.



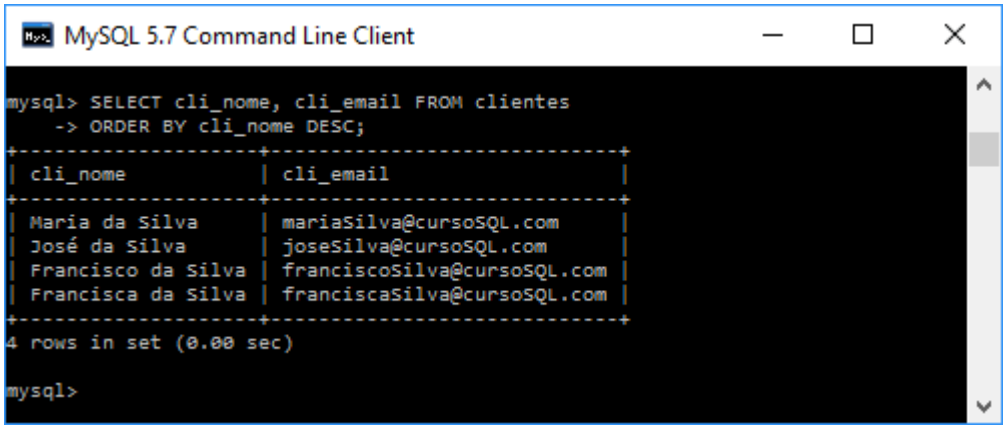
Fonte: MySQL 5.7 *Command Line Client*

O MySQL sempre apresenta os resultados em ordem ascendente, ou seja, se estivermos tratando de dados de texto, na ordem de “A” a “Z”. Se desejarmos visualizar em ordem inversa, adicionamos o termo DESC na cláusula ORDER BY, como apresentado no quadro a seguir.

```
mysql>SELECT  cli_nome, cli_email
   FROM clientes
   ORDER BY cli_nome DESC;
```

Observe que agora as informações sobre os nomes e e-mails são apresentadas em ordem alfabética inversa de acordo com o nome do cliente, conforme é ilustrado na **Figura 3**.

Figura 03 - Tela do MySQL após o comando **SELECT cli_clientes, cli_email FROM clientes ORDER BY cli_nome DESC**.

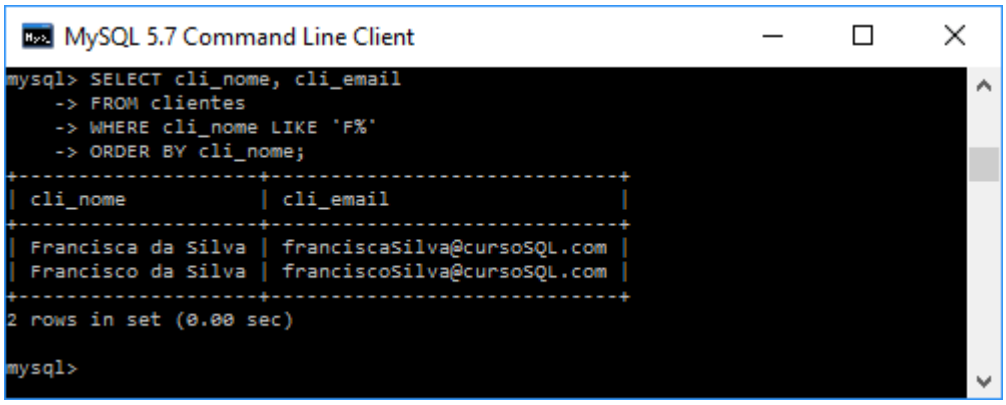


Fonte: MySQL 5.7 *Command Line Client*

A cláusula ORDER BY pode ser utilizada em conjunto com a cláusula WHERE, de modo a visualizarmos os dados que atendam uma determinada condição de forma ordenada. Esse tipo de consulta deve seguir a sequência: SELECT ... FROM ... WHERE ... ORDER BY .... Por exemplo, se quisermos o nome e e-mail dos clientes cujos nomes começam com a letra "F" de forma ordenada, utilizamos o SELECT como apresentado no quadro a seguir, obtendo o resultado ilustrado na **Figura 4**.

```
1 mysql>SELECT  cli_nome, cli_email
2 FROM clientes
3 WHERE cli_nome LIKE 'F%'
4 ORDER BY cli_nome;
```

Figura 04 - Tela do MySQL após o comando **SELECT cli_clientes, cli_email FROM clientes WHERE cli_nome LIKE 'F%' ORDER BY cli_nome**.



Fonte: MySQL 5.7 *Command Line Client*

Vídeo 02 - Ordenação de Resultados

Atividade 01

1. Elabore os comandos SQL para realizar as seguintes consultas no banco de dados da nossa locadora:
    a. Exibir nome e CPF de todos os clientes em ordem alfabética;
    b. Exibir todas as informações sobre os filmes da locadora em ordem decrescente do código do filme;
    c. Exibir o nome e-mail dos clientes do sexo masculino em ordem alfabética inversa;
    d. Exibir o nome e data de nascimento dos clientes que usam e-mail do Hotmail (@hotmail.com) em ordem alfabética.

Consultas usando agrupamento de dados

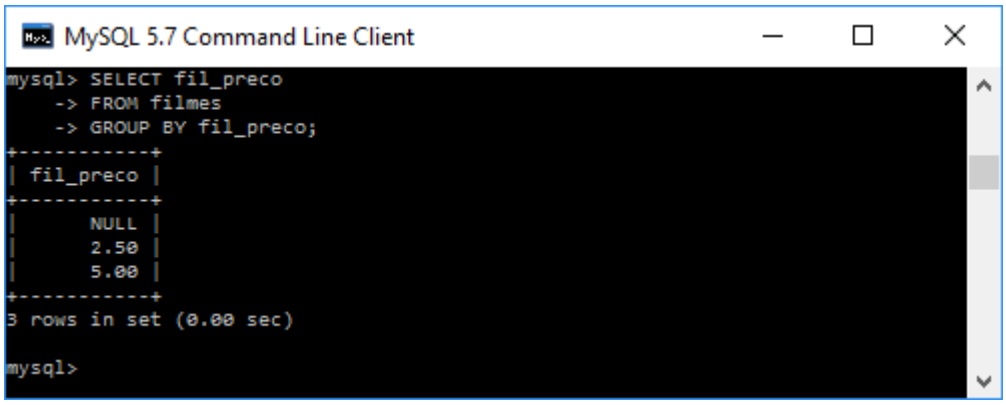
Existem situações em que precisamos fazer consultas considerando algum tipo de agrupamento das linhas de uma tabela. Por exemplo: quantos clientes do sexo feminino têm na locadora? Quantos exemplares de um mesmo filme estão cadastrados na locadora? Essas perguntas envolvem algum tipo de categorização dos dados. Na primeira, é necessário separar os clientes do sexo masculino dos clientes do sexo feminino e na segunda os registros de um mesmo título. Ou seja, seria muito útil que tivéssemos algum mecanismo de criar categorias internas com os dados disponíveis sem a necessidade de modificar a estrutura da tabela.

Para resolver isso, o comando SELECT tem a cláusula **GROUP BY**, que agrupa as linhas com valores iguais de uma determinada coluna. Meio confuso? Um exemplo nos ajudará a entender. Imagine que precisamos saber os diferentes preços de uma locação na locadora, nessa situação utilizamos o comando SELECT, conforme apresentado no quadro a seguir.

```
mysql>SELECT  fil_preco
   FROM filmes
   GROUP BY fil_preco;
```

Observe que o resultado da consulta nos informa que temos filmes sem preço cadastrado (NULL) e filmes que custam R$2,50 ou R$5, conforme ilustrado na **Figura 5**. Como foi gerado esse resultado? O GROUP BY agrupou todos os filmes que tinham mesmo valor cadastrado na coluna *fil_preco* como se fosse um registro único e mostrou na tela os diferentes valores presentes na coluna *fil_preco* segundo esse agrupamento.

Figura 05 - Tela do MySQL após o comando **SELECT fil_preco FROM filmes GROUP BY fil_preco**.



Fonte: MySQL 5.7 *Command Line Client*

O principal uso da cláusula GROUP BY ocorre em conjunto com funções de agregação. As funções de agregação retornam um único valor como resultado de um conjunto de valores de entrada. As principais funções de agregação usadas no *My*SQL são:

- **COUNT()**: Função que retorna a quantidade total de registros não-nulos de um atributo;
- **SUM()**: Função que retorna a soma dos valores de um atributo;
- **AVG()**: Função que retorna a média dos valores de um atributo;
- **MIN()**: Função que retorna o menor valor de um atributo;
- **MAX()**: Função que retorna o maior valor de um atributo.

Exemplos

Para entendermos melhor a utilização dessas funções, vamos analisar os seguintes exemplos.

Exemplo 1

Calcular o número total de filmes cadastrados em nosso banco de dados.

```
mysql>SELECT COUNT(fil_titulo)
   FROM filmes;
```

Exemplo 2

Mostrar quantos títulos diferentes de filmes estão cadastrados em nosso banco de dados.

```
mysql>SELECT  COUNT(DISTINCT fil_titulo)
   FROM filmes;
```

A cláusula DISTINCT elimina as linhas duplicadas que aparecerão na consulta, ou seja, valores repetidos de um atributo são considerados apenas uma vez.

Exemplo 3

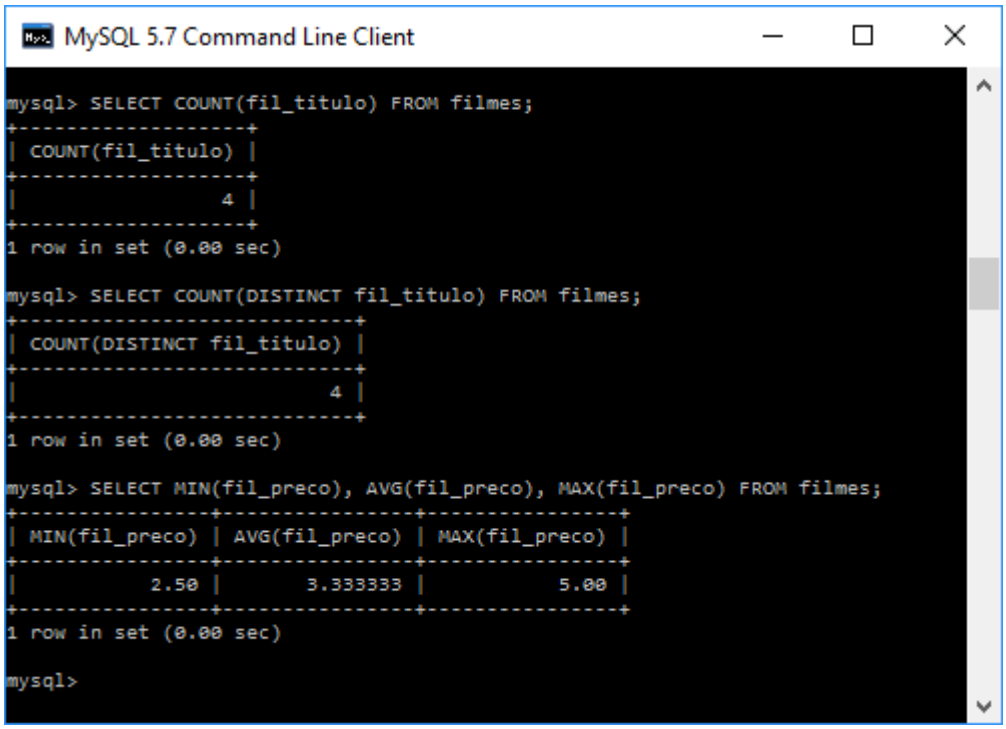
Calcular os valores mínimo, médio e máximo de uma locação em nossa locadora.

```
mysql>SELECT  MIN(fil_preco), AVG(fil_preco), MAX(fil_preco)
   FROM filmes;
```

As funções SUM() e AVG() são utilizadas apenas em atributos do tipo numérico.

Os resultados dessas pesquisas são ilustrados na **Figura 6**.

Figura 06 - Tela do MySQL após diversas pesquisas com o comando SELECT, usando funções de agregação.



Fonte: MySQL 5.7 *Command Line Client*

Vídeo 03 - Funções de Agregação

Consultas usando agrupamento de dados - pt.2

Agora que conhecemos as funções de agregação, vamos fazer uso delas em conjunto com a cláusula GROUP BY. Vamos resolver as questões que levantamos no início da nossa discussão.

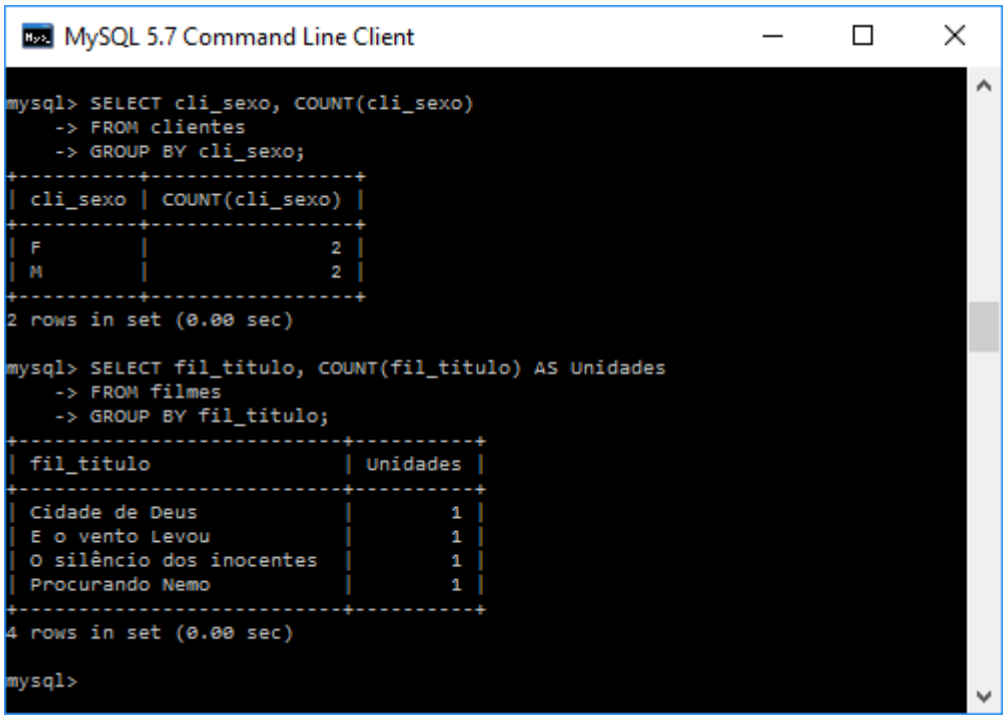
1. Quantos clientes do sexo feminino têm na locadora?
2. Quantos exemplares de um mesmo filme estão cadastrados na locadora?

A solução dessas questões é apresentada no quadro a seguir.

```
mysql>SELECT  cli_sexo, COUNT(cli_sexo)
FROM clientes
GROUP BY cli_sexo;
mysql>SELECT  fil_titulo, COUNT(fil_titulo) AS Unidades
FROM filmes
GROUP BY fil_titulo;
```

Na primeira consulta, agrupamos os dados de acordo com o sexo do cliente e contamos quantos registros estão associados a cada categoria, no caso, Masculino (M) e Feminino (F). Na segunda consulta, agrupamos os dados de acordo com o título do filme e contamos quantos registros existiam para cada título. O resultado é ilustrado na **Figura 7**.

Figura 07 - Tela do MySQL após diversas pesquisas com o comando SELECT, usando funções de agregação junto com a cláusula GROUP BY.



Fonte: MySQL 5.7 *Command Line Client*

Você percebeu algo diferente na segunda consulta ilustrada no exemplo anterior? Apareceu um termo diferente na expressão “COUNT(fil_titulo) AS Unidades”. Esse termo AS é uma cláusula utilizada no SELECT para renomear o nome de uma coluna de resultados e pode ser usado para deixar a consulta mais legível. No exemplo indicado, ao invés de aparecer no resultado uma coluna intitulada COUNT(fil_titulo), apareceu uma coluna intitulada Unidades, que torna mais fácil o entendimento do resultado da pesquisa. Podemos dizer que demos um apelido para a coluna apresentada.

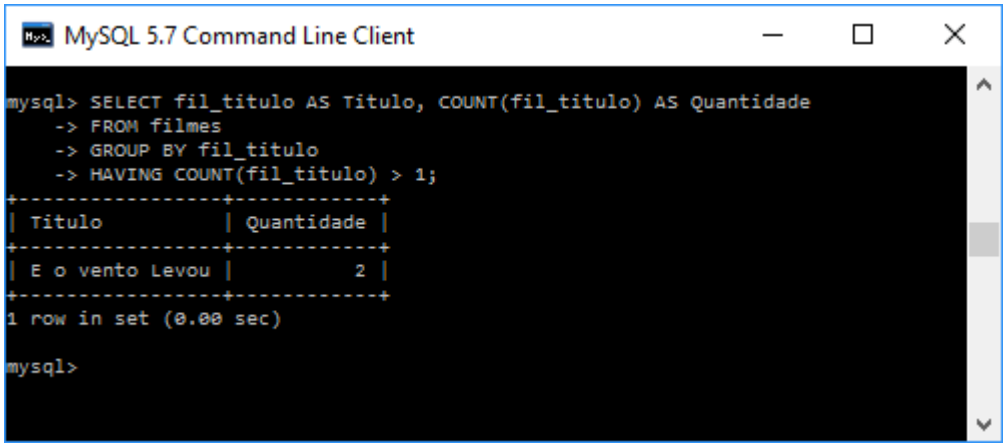
Quando fazemos uma pesquisa num agrupamento de dados, é possível restringir os dados que serão agrupados de acordo com alguma condição. Para isso, utilizamos a cláusula HAVING em conjunto com GROUP BY. Por exemplo, considere a seguinte questão: dentre os filmes cadastrados na locadora, quais possuem mais de uma unidade? A solução é apresentada no quadro a seguir.

Você deve ter notado que nosso banco de dados da locadora possui apenas 4 filmes distintos (E o vento Levou, O silêncio dos inocentes, Procurando Nemo e Cidade de Deus). Para que o exemplo abaixo apresente duas unidades do filme "E o vento Levou" será necessário inserir uma nova cópia deste filme no banco de dados. É importante que o título do filme seja exatamente "E o vento Levou" já inserido anteriormente. Se tiver dúvidas, consulte o seu tutor.

```
mysql>SELECT fil_titulo AS Titulo, COUNT(fil_titulo) AS Quantidade
   FROM filmes
   GROUP BY fil_titulo
   HAVING COUNT(fil_titulo)>1;
```

Nesse caso, são apresentados os resultados apenas dos filmes que após a classificação pelo título apresentam mais de uma unidade cadastrada, conforme ilustrado na **Figura 8**.

Figura 08 - Tela do MySQL após o comando SELECT, junto com as cláusulas GROUP BY e HAVING.



Fonte: MySQL 5.7 *Command Line Client*

Atividade 02

1. Elabore os comandos SQL para realizar as seguintes consultas no banco de dados da nossa locadora:

    a. Exibir a quantidade de filmes alugados e de filmes disponíveis na locadora;

    b. Exibir a quantidade de filmes de cada gênero cadastrados na locadora;

    c. Calcular a quantidade total de dinheiro a ser recebida pela locadora quando os clientes devolverem os filmes que estão alugados, considerando que eles pagam apenas na devolução do filme;

    d. Modificar as consultas anteriores renomeando as colunas de resultados de modo que o resultado fique mais legível.

Consultas usando funções especiais

Além das funções de agregação discutidas na seção anterior, existem outras funções que podem ser usadas no comando SELECT. As principais são:

1. **Funções matemáticas:** definição de operações matemáticas avançadas;
2. **Funções de manipulação de** *strings:* usadas na manipulação de dados do tipo caractere;
3. **Funções de data/hora:** usadas na manipulação de dados do tipo DATE e TIME.

O **Quadro 1**, o **Quadro 2** e o **Quadro 3** apresentam um resumo de algumas funções de cada uma das categorias apresentadas na lista anterior.

Função	Significado
ABS (*valor*)	Retorna o valor absoluto (positivo) do valor informado
FLOOR (*valor*)	Retorna o maior número inteiro, igual ou menor ao valor informado
ROUND (*valor*, **n**)	Arredonda o valor informado para **n** casas decimais
POWER (*valor*, **p**)	Retorna o valor informado elevado à potência **p**

Quadro 1 – Exemplos de funções matemáticas

Função	Significado
LENGTH (*expressão*)	Retorna o número de caracteres contidos na expressão informada
LOWER (*expressão*) e UPPER (*expressão*)	Converte para minúsculo e maiúsculo a expressão informada, respectivamente
LTRIM (*expressão*) e RTRIM (*expressão*)	Remove os espaços em branco à esquerda e à direita da expressão informada,respectivamente
SUBSTRING (*expressão, início, tamanho*):	Extrai uma parte dos caracteres da expressão, iniciando da posição informada em início, considerando a quantidade definida em tamanho

Quadro 2 – Exemplos de funções de manipulação de *strings*

Função	Significado
CURDATE() e CURTIME()	Retorna a data e hora atuais, respectivamente
EXTRACT (*parte* FROM *data*)	Retorna apenas a parte especificada de um campo de data/hora. A parte pode ser *year*, *month*, *day*, *hour*, *minute* etc.
DATE FORMAT (*data, formato*)	Retorna a data modificando seu formato de apresentação, que pode ser: %d para dia (0-31), %m para mês (0-12), %Y para ano com quatro dígitos etc.

Quadro 3 – Exemplos de funções de data/hora

Exemplos

Para entendermos melhor a utilização dessas funções, vamos analisar os seguintes exemplos.

Exemplo 1

Mostrar o preço médio das locações considerando apenas duas casas decimais.

```
mysql>SELECT DISTINCT ROUND(AVG(fil_preco),2)
   FROM filmes;
```

Exemplo 2

Mostrar em MAIÚSCULO os títulos de todos os filmes (sem repetição) que estão alugados.

```
mysql>SELECT DISTINCT UPPER(fil_titulo)
   FROM filmes
   WHERE fil_situacao='alugado';
```

Exemplo 3

Mostrar o nome e ano de nascimento de todos os clientes da locadora do sexo feminino.

```
mysql>SELECT cli_nome, EXTRACT(year FROM cli_data_nasc)
   FROM clientes
   WHERE cli_sexo='F';
```

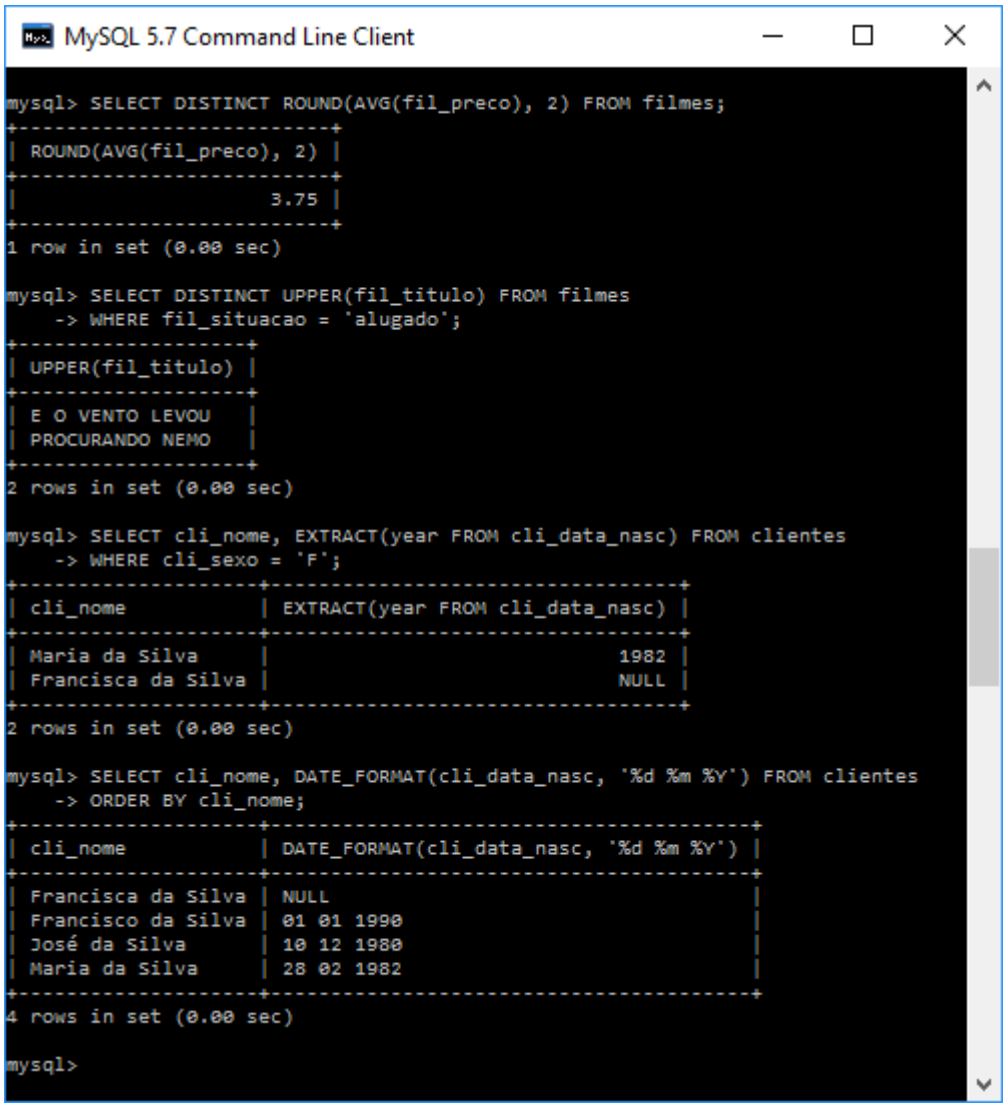
Exemplo 4

Mostrar o nome e data de nascimento de todos os clientes da locadora no formato brasileiro dd/mm/aaaa e em ordem alfabética do nome.

```
mysql>SELECT cli_nome, DATE_FORMAT(cli_data_nasc,'%d %m %Y')
   FROM clientes
   ORDER BY cli_nome;
```

Os resultados dessas pesquisas são ilustrados na **Figura 9**.

Figura 09 - Tela do MySQL após diversas pesquisas com o comando SELECT, usando funções especiais.



Fonte: MySQL 5.7 *Command Line Client*

Vídeo 04 - Funções Especiais

Atividade 03

1. Nesta aula, foram apresentadas algumas funções especiais que podemos usar em conjunto com o comando SELECT. Faça uma busca na internet e descubra outros tipos de funções definidas pelo MySQL.

Conclusão

Encerramos por aqui nossa quarta aula sobre a linguagem SQL. Na próxima, aprenderemos a modificar a estrutura de uma tabela, criando novas colunas ou redefinindo colunas já existentes e mostraremos como usar a linguagem SQL em banco de dados com múltiplas tabelas. Você deve ter observado que o comando SELECT é muito poderoso e possui diversas formas de uso.

Para ajudar na fixação do conteúdo, pegue seu caderno e faça um resumo do que já foi estudado sobre ele e exercite bastante para não esquecer. Lembre-se de fazer sua autoavaliação. Bons estudos e boa sorte!

Resumo

Nesta aula, você deu continuidade ao estudo sobre o comando SELECT e estudou as cláusulas ORDER BY e GROUP BY que são utilizadas para ordenar e agrupar os dados pesquisados, respectivamente. Em relação ao agrupamento, você viu que podemos refinar os resultados agrupados usando a cláusula HAVING. Viu também que as funções de agregação COUNT(), SUM(), AVG(), MIN() e MAX() ajudam a obter novas informações sobre nossos dados, sobretudo, quando unidas ao GROUP BY. Conheceu cláusulas que ajudam a melhorar a apresentação dos nossos resultados como DISTINCT e AS. Viu também diversas funções especiais que tratam tipos numéricos, caracteres e data/hora que podem nos auxiliar a fazer pesquisas mais complexas ou melhorar a apresentação dos nossos resultados.

Autoavaliação

1. Considere o banco de dados CursoX criado nas aulas anteriores cuja estrutura de tabelas é mostrada a seguir:

ATRIBUTO	TIPO	DESCRIÇÃO
aluno_cod	Número inteiro	Código do aluno
aluno_nome	Alfanumérico	Nome do aluno
aluno_endereco	Alfanumérico	Endereço do aluno
aluno_cidade	Alfanumérico	Cidade do aluno

TABELA: Alunos

ATRIBUTO	TIPO	DESCRIÇÃO
dis_cod	Número inteiro	Código da disciplina
dis_nome	Alfanumérico	Nome da disciplina
dis_carga	Número inteiro	Carga horária da disciplina
dis_professor	Alfanumérico	Professor da disciplina

TABELA: Disciplina

ATRIBUTO	TIPO	DESCRIÇÃO
prof_cod	Número inteiro	Código do professor
prof_nome	Alfanumérico	Nome do professor
prof_endereco	Alfanumérico	Endereço do professor
prof_cidade	Alfanumérico	Cidade do professor

TABELA: Professores

Resolva as consultas utilizando a linguagem SQL.

a. Exiba as diferentes cidades em que moram os alunos e as respectivas quantidades de alunos em cada uma.

b. Refaça a consulta do item a) para os professores, ordenando o resultado em ordem alfabética de acordo com o nome da cidade e renomeando as colunas de resultados.

c. Exiba a quantidade total de disciplinas oferecidas.

d. Exiba a carga horária total para cada professor.

e. Exiba a carga horária total de cada professor que possua mais de uma disciplina.

f. Exiba o nome de todas as disciplinas ofertadas (sem repetição de nome) em MAIÚSCULO.

g. Exiba o nome de todos os professores cujos nomes iniciem com as letras de “A” a “J” em ordem alfabética.

Calcule a carga horária média por professor (dica: Você pode fazer mais de uma consulta para obter esse resultado).

Referências

BEIGHLEY, L. **Use a cabeça SQL**. Rio de Janeiro: Alta Books, 2008.

MYSQL 5.7 **Reference Manual**. Disponível em: <http://dev.mysql.com/doc/refman/5.7/en/>. Acesso em: 15 jan. 2017.