MD.UnidadDidácticaGrupo(01)Pt

MÓDULO

BASE DE DADOS MYSQL

UNIDADE

RECUPERAÇÃO DE DADOS EM MYSQL

UDD10000_V(01)

Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Saber reconhecer as consultas de seleção de dados do SQL.
- Saber reconhecer algumas das próprias funções do MySQL.
- Conseguir fazer pesquisas ordenadas.
- Aprender a pesquisar com filtros para determinadas condições.
- Aprender a fazer pesquisas agrupando os resultados de acordo com determinados critérios.
- Saber pesquisar entre tabelas relacionadas.

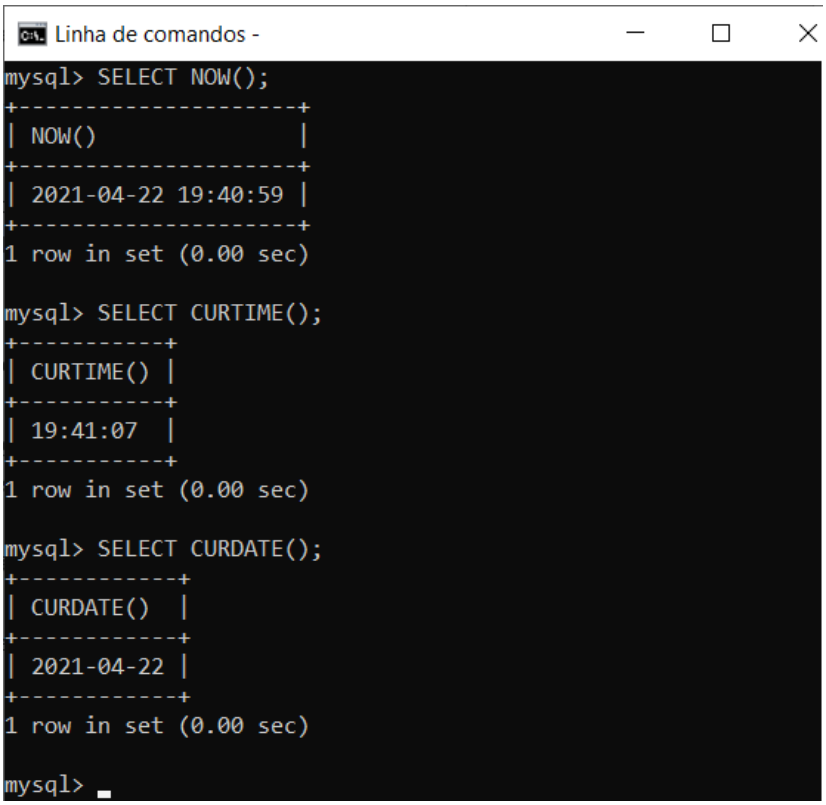
INTRODUÇÃO

Todas as informações recolhidas com antecedência devem ser processadas adequadamente para que sejam significativas e úteis. A maneira de obter respostas na vida real é fazer perguntas, mais especificamente, se pretender as melhores informações, fazer as perguntas certas às pessoas certas. Há momentos em que ”questionamos” objetos inanimados, como um dicionário ou uma enciclopédia; nesses casos, o verbo “consultar” pode ser mais apropriado, embora o objetivo seja o mesmo: obter informações que sejam necessárias no momento da pergunta.

Depois de estudar esta unidade didática será capaz de fazer essas "perguntas" ou consultas à base de dados de futebol e obter algumas respostas. Vai perceber como é fácil obter respostas para problemas que podem ser quantificados, como os resultados desportivos.

1. FUNÇÕES MYSQL

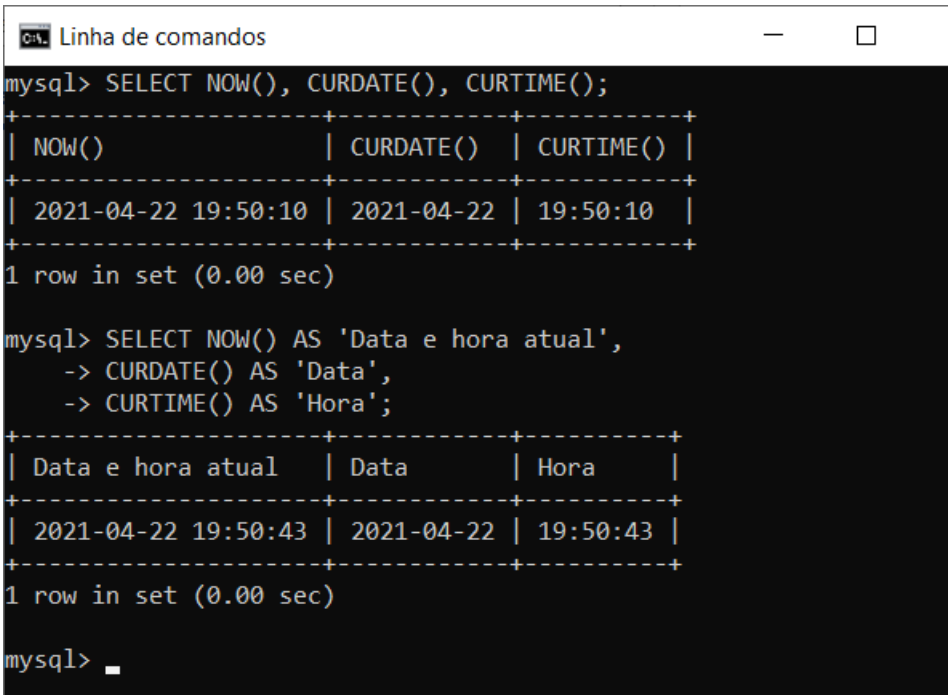
Esta unidade didática começa pela função mais comum: a função responsável por devolver a data atual, que está até presente em qualquer eletrodoméstico. Desta forma, poderá saber se o computador onde o MySQL está instalado tem a mesma hora do seu relógio.



Como pode ver na imagem anterior, para saber a data e hora atuais basta executar **SELECT NOW();**. Se pretender saber apenas a hora deverá executar SELECT CURTIME();. Se pretender saber apenas a data, deverá executar SELECT CURDATE();.

A seguir é apresentada uma adaptação do exemplo anterior, onde é possível reunir as três frases anteriores numa só; basta escrever um SELECT e, a seguir, separadas por vírgulas, as funções de tempo utilizadas no exemplo anterior. Como poderá ver na imagem seguinte, desta forma foi possível devolver os resultados numa única tabela e em três colunas diferentes. Por fim, e numa versão atualizada, obteve-se o mesmo resultado, mas com uma linha com os títulos das colunas desejados. Para isso, escreveu-se cada título após a palavra "AS" e entre plicas.

```
SELECT NOW(), CURDATE(), CURTIME();
SELECTNOW() AS 'Data e hora atual',
CURDATE() AS 'Data',
CURTIME() AS 'Hora';
```



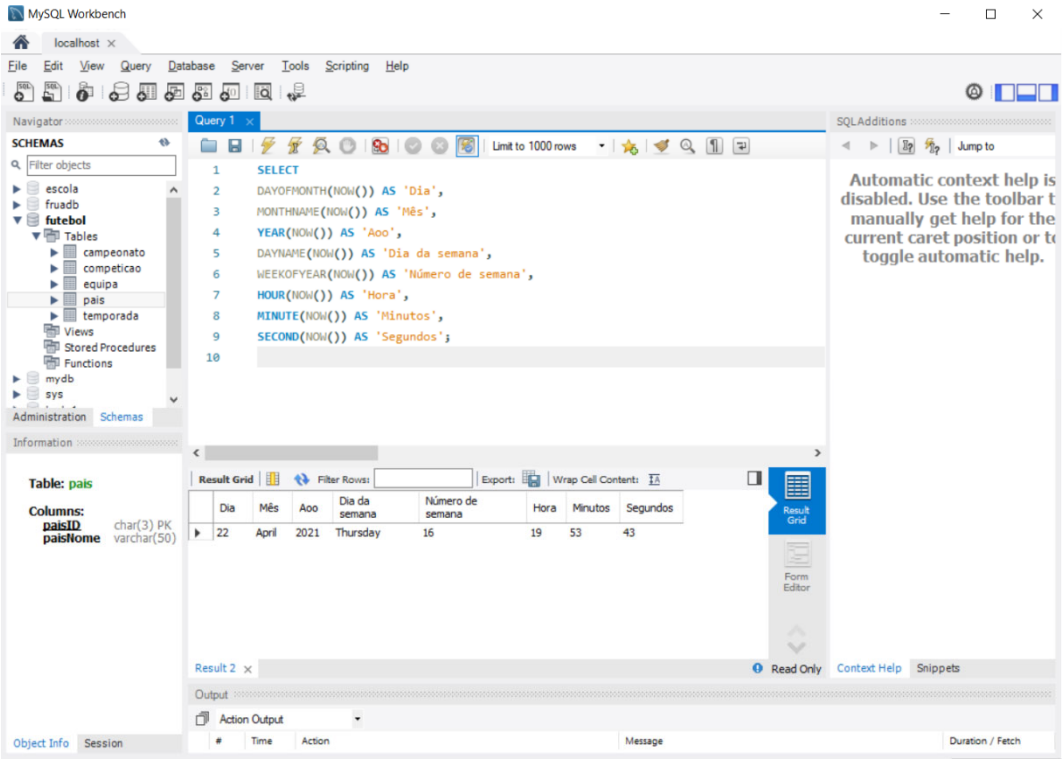
Todas estas funções e outras para trabalhar com datas e horas podem ser encontradas em http://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html. Olhando para estes exemplos, reparará que são fáceis de aprender e que precisa apenas de fazer alguns testes para obter os resultados desejados. A seguir, é apresentada uma lista e, por fim, um exemplo bastante ilustrativo.

- **ADDDATE()**: adiciona um intervalo de tempo a uma data.
- **ADDTIME()**: adiciona tempo.
- **CONVERT_TZ()**: permite fazer conversões de um fuso horário para outro.
- **CURDATE(), CURRENT_DATE(), CURRENT_DATE**: devolve a data atual.
- **CURTIME(), CURRENT_TIME(), CURRENT_TIME**: devolve a hora, minutos e segundos na hora atual.
- **DATE_ADD()**: permite adicionar valores a uma data.
- **DATE_FORMAT()**: permite fornecer um formato específico a uma data.
- **DATE_SUB()**: permite subtrair valores de uma data.
- **DATE()**: permite extrair a data de uma expressão temporária.
- **DATEDIFF()**: permite obter a subtração de duas datas.
- **DAYNAME()**: devolve o nome do dia da semana.
- **DAYOFMONTH(), DAY()**: devolve o dia do mês.
- **DAYOFWEEK()**: devolve o índice do dia da semana, sendo 1 domingo.
- **DAYOFYEAR()**: devolve o dia do ano (de 1 a 366).
- **EXTRACT()**: permite extrair uma parte de uma data.
- **FROM_DAYS()**: dado um número de dias, converte-o numa data.
- **FROM_UNIXTIME()**: dada uma data no formato UNIX, devolve no formato escolhido ou da forma tradicional se nenhum for indicado.
- **GET_FORMAT()**: devolve uma string com o formato indicado para a data.
- **HOUR()**: permite extrair a hora de uma data.
- **LAST_DAY**: devolve o último dia do mês da data fornecida.
- **MAKEDATE()**: permite criar uma data do ano e do dia do ano.

- **MAKETIME()**: cria uma hora a partir de horas, minutos e segundos.
- **MICROSECOND()**: devolve a parte correspondente aos microssegundos da data fornecida.
- **MINUTE()**: devolve os minutos da data fornecida.
- **MONTH()**: devolve o número do mês para a data fornecida.
- **MONTHNAME()**: devolve o nome do mês para a data fornecida.
- **NOW(), CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP,LOCALTIME(), LOCALTIME, LOCALTIMESTAMP,LOCALTIMESTAMP():** devolve a data e a hora atuais.
- **PERIOD_ADD()**: permite adicionar um certo número de meses a um período.
- **PERIOD_DIFF()**: permite subtrair dois períodos e assim saber o número de meses de diferença.
- **QUARTER():** devolve o número do trimestre de uma data.
- **SEC_TO_TIME()**: converte os segundos em formato de hora, minuto e segundo.
- **SECOND()**: devolve a parte correspondente aos segundos de uma data (0-59).
- **STR_TO_DATE()**: converte uma sequência de caracteres numa data.
- **SUBDATE()**: é sinónimo de DATE_SUB() quando invocado com três argumentos.
- **SUBTIME()**: permite subtrair horas.
- **SYSDATE()**: devolve a hora em que uma função é executada.
- **TIME_FORMAT()**: permite formatar as horas.
- **TIME_TO_SEC()**: devolve o número de segundos numa expressão de tempo.
- **TIME()**: extrai a parte dedicada às horas, minutos e segundos de uma determinada data.
- **TIMEDIFF()**: permite subtrair os tempos.
- **TIMESTAMP()**: com um único argumento, devolve a data; com dois argumentos devolve a soma das datas.
- **TIMESTAMPADD()**: adiciona um intervalo a uma data.

- **TIMESTAMPDIFF()**: permite subtrair um intervalo de uma data.
- **TO_DAYS()**: devolve a data convertida em dias.
- **UNIX_TIMESTAMP()**: devolve uma data no formato UNIX.
- **UTC_DATE()**: devolve a data no formato UTC.
- **UTC_TIME()**: devolve a hora no formato UTC.
- **UTC_TIMESTAMP()**: devolve a data e hora no formato UTC.
- **WEEK()**: devolve o número da semana.
- **WEEKDAY()**: devolve o índice do dia da semana, a começar no 0 para segunda-feira.
- **WEEKOFYEAR()**: devolve o número da semana do ano (de 0 a 53).
- **YEAR()**: devolve o ano de uma data.
- **YEARWEEK()**: devolve o ano e a semana de uma data.

```
SELECTDAYOFMONTH(NOW()) AS 'Dia',
MONTHNAME(NOW()) AS 'Mês',
YEAR(NOW()) AS 'Aoo',
DAYNAME(NOW()) AS 'Dia da semana',
WEEKOFYEAR(NOW()) AS 'Número de semana',
HOUR(NOW()) AS 'Hora',
MINUTE(NOW()) AS 'Minutos',
SECOND(NOW()) AS 'Segundos';
```



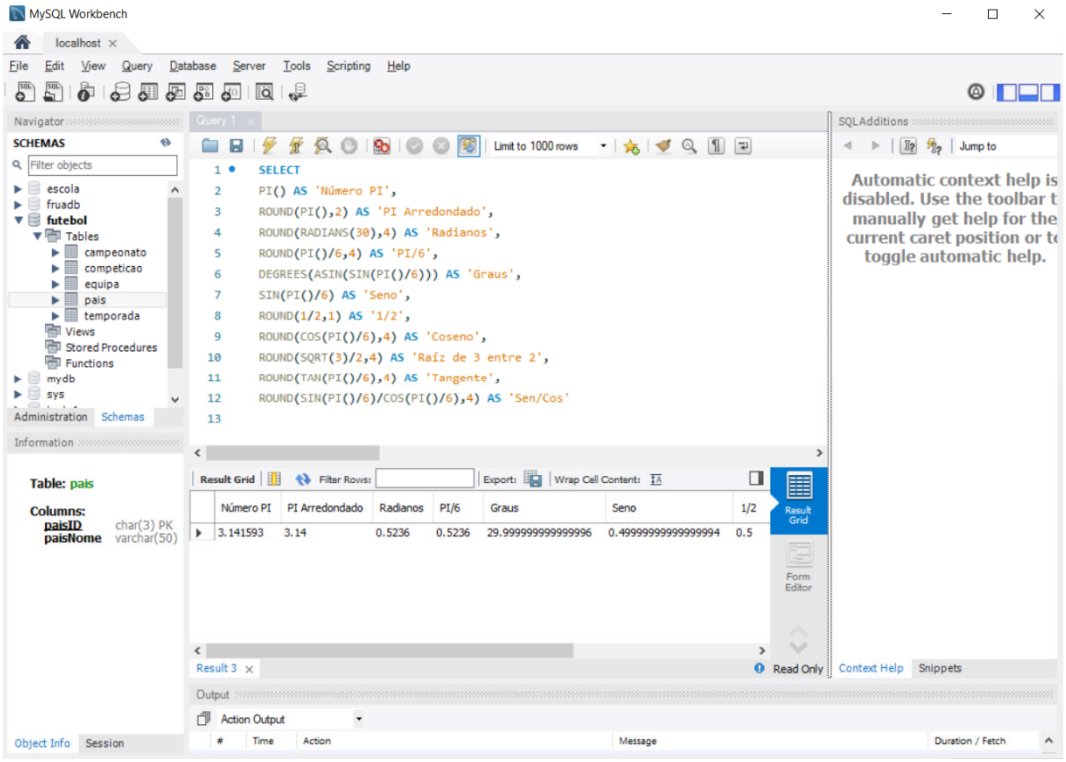
Por enquanto, trabalhou com funções do tipo data e hora, mas existem muitas outras funções úteis que pode usar, como as funções de string ou as funções que permitem trabalhar com números. Na página web http://dev.mysql.com/doc/refman/5.1/en/numeric-functions.html, pode encontrar as principais funções e operadores matemáticos, também presentes na lista seguinte.

- **ABS()**: devolve o valor absoluto.
- **ACOS()**: devolve o arco cosseno.
- **ASIN()**: devolve o arco seno.
- **ATAN2(), ATAN()**: devolve o arco tangente de dois argumentos, calculando efetivamente o arco tangente da divisão de ambos, tendo em consideração os seus sinais para poder discernir o quadrante correspondente.
- **ATAN()**: devolve o arco tangente.
- **CEIL(), CEILING()**: devolve o menor inteiro maior do que o valor fornecido como argumento.
- **CONV()**: converte números de diferentes bases, com a menor base sendo 2 e a maior sendo a base 36.

- **COS()**: devolve o cosseno do número recebido em radianos.
- **COT()**: devolve a cotangente.
- **CRC32()**: calcula o valor de verificação de redundância cíclica.
- **DEGREES()**: converte radianos em graus.
- **DIV**: divisão inteira.
- **/**: operador de divisão.
- **EXP()**: devolve o valor do número e elevado ao valor do argumento fornecido.
- **FLOOR()**: devolve o maior dos valores inteiros, menor do que o argumento fornecido.
- **LN()**: devolve o logaritmo natural ou natural da base e.
- **LOG10()**: devolve o logaritmo para a base 10.
- **LOG2()**: devolve o logaritmo para a base 2.
- **LOG()**: devolve o logaritmo da base e o número fornecido; se a base não for fornecida, o logaritmo natural é devolvido.
- **-**: operador de subtração.
- **MOD()**: devolve o resto da divisão inteira.
- **%**: operador que devolve o restante da divisão inteira.
- **OUT()**: devolve a representação octal de um número decimal.
- **PI()**: devolve o valor do número π.
- **+**: operador de adição.
- **POW(), POWER()**: devolve o valor de um número elevado a uma determinada potência.
- **RADIANS()**: converte um determinado ângulo em graus para radianos.
- **RAND()**: devolve um número aleatório decimal entre 0 e 1.
- **ROUND()**: arredonda um número para o número especificado de casas decimais.
- **SIGN()**: devolve o sinal do argumento, sendo -1 se for negativo, +1 se for positivo e 0 se for 0.
- **SIN()**: devolve o seno de um determinado ângulo em radianos.

- **SQRT()**: devolve a raiz quadrada de um número.
- **SO()**: devolve a tangente de um determinado ângulo em radianos.
- ***: operador de multiplicação.
- **TRUNCATE()**: devolve um número truncado em certas casas decimais.
- **-**: permite alterar o sinal de um argumento.

Em seguida, verá um exemplo de algumas funções e operadores anteriormente apresentados aplicados à trigonometria, mais especificamente, para trabalhar com um ângulo de 30°, que é um dos ângulos mais representativos na matemática, e que corresponde a π/6 radianos (conforme demonstrado aplicando a consulta seguinte).



```
SELECT
PI() AS 'Número PI',
ROUND(PI(),2) AS 'PI Arredondado',
ROUND(RADIANS(30),4) AS 'Radianos',
ROUND(PI()/6,4) AS 'PI/6',
DEGREES(ASIN(SIN(PI()/6))) AS 'Graus',
SIN(PI()/6) AS 'Seno',
ROUND(1/2,1) AS '1/2',
ROUND(COS(PI()/6),4) AS 'Cosseno',
ROUND(SQRT(3)/2,4) AS 'Raiz de 3 entre 2',
ROUND(TAN(PI()/6),4) AS 'Tangente',
ROUND(SIN(PI()/6)/COS(PI()/6),4) AS 'Sen/Cos'
```

Falta então ver as funções que podem ser aplicadas a textos e strings. Funções que permitem, entre outras coisas, concatenar strings, converter para maiúsculas e minúsculas, eliminar espaços em branco, inverter a ordem das letras, etc. Também poderá ler mais sobre as mesmas na seguinte página web: http://dev.mysql.com/doc/refman/5.1/en/string-functions.html. Segue-se uma lista com as principais:

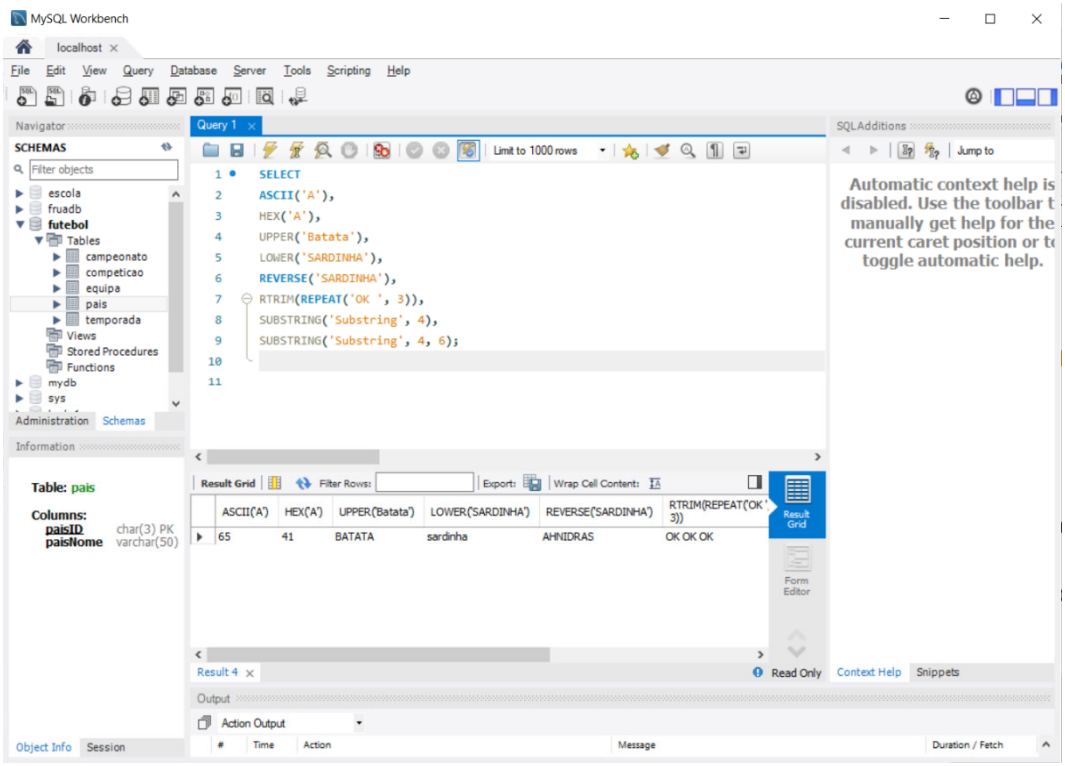
- **ASCII()**: devolve o valor numérico ASCII do primeiro carácter.
- **BIN()**: devolve o valor binário de uma string contendo um número decimal.
- **BIT_LENGTH()**: devolve o comprimento da string em bits.
- **CHAR_LENGTH(), CHARACTER_LENGTH()**: devolve o número de caracteres na string.
- **CHAR()**: devolve o carácter correspondente ao número ASCII fornecido como argumento.
- **CONCAT_WS()**: une as strings passadas como parâmetros com o separador especificado.
- **CONCAT()**: une as strings passadas como parâmetros.
- **ELT()**: devolve uma string de várias possíveis, dependendo do índice fornecido.

- **EXPORT_SET()**: devolve uma string em que, para cada bit do valor dos bits, pode obter uma string on e, para cada bit reatribuído, uma string off. Os bits são verificados da direita para a esquerda (do mais baixo para o mais alto). As strings são adicionadas ao resultado da esquerda para a direita, separadas pelo separador de string (o carácter padrão é a vírgula). O número de bits examinados é obtido pelo parâmetro number_of_bits (por padrão 64).
- **FIELD()**: função que, ao passar uma string, procura correspondências noutras strings, e devolve a posição correspondente à string que corresponde totalmente. Devolve 0 caso não encontre nenhuma.
- **FIND_IN_SET()**: dada uma string e uma lista de strings, devolve a posição da lista de strings a que corresponde.
- **FORMAT()**: formata um número com o formato '#, ###, ###. ##' e o número especificado de casas decimais, devolvendo o resultado como uma sequência de caracteres.
- **HEX()**: devolve a representação hexadecimal de um valor decimal ou de uma string.
- **INSERT()**: permite inserir uma string na outra a partir da posição indicada e com o comprimento previsto.
- **INSTR()**: devolve a posição da primeira correspondência de uma string dentro de outra.
- **LEFT()**: devolve os primeiros x caracteres de uma string.
- **LENGTH(), OCTET_LENGTH()**: devolve o comprimento da string em bytes.
- **LIKE**: permite comparar strings com um padrão.
- **LOAD_FILE()**: permite ler um ficheiro e carregar o seu conteúdo como uma string.
- **LOCATE(), POSITION()**: permite localizar uma string dentro de outra.
- **LOWER(), LCASE()**: devolve a string em minúsculas.
- **LPAD()**: permite adicionar outra string à esquerda de uma string que se repetirá quantas vezes forem necessárias para que a string alcance o comprimento final marcado.
- **LTRIM()**: exclui os espaços à esquerda da string.

- **MAKE_SET()**: devolve um conjunto de valores separados por vírgulas, que consiste em strings que possuem o bit em bits correspondente atribuído. Str1 corresponde ao bit 0, str2 ao bit 1 e assim por diante. Valores NULL em str1, str2... não são adicionados ao resultado.
- **MATCH**: permite pesquisar por texto.
- **NOTLIKE**: negação do LIKE.
- **NOT REGEXP**: negação de REGEXP.
- **ORD()**: devolve o código do primeiro carácter.
- **QUOTE()**: limita uma string para produzir um resultado que pode ser usado como um valor de escape num comando SQL.
- **REGEXP, RLIKE**: permite combinar um padrão utilizando expressões regulares.
- **REPEAT()**: repete uma sequência de caracteres por um determinado número de vezes.
- **REPLACE()**: substitui as ocorrências de uma string por outra fornecida como parâmetro.
- **REVERSE()**: devolve uma string com a ordem dos seus caracteres invertida.
- **RIGHT()**: devolve os últimos x caracteres de uma string.
- **RPAD()**: muito parecido com o LPAD, mas, neste caso, a string que se repete quantas vezes forem necessárias para completar o comprimento ficará à direita.
- **RTRIM()**: devolve a string sem espaços à direita da string.
- **SOUNDEX()**: devolve uma stringsoundex. Duas cordas com o mesmo som vão devolver o mesmo resultado.
- **SOUNDSLIKE**: permite comparar sons.
- **SPACE()**: devolve uma string com x espaços.
- **STRCMP()**: permite comparar duas strings.
- **SUBSTRING_INDEX()**: devolve uma substring antes de um número especificado de ocorrências do delimitador.
- **SUBSTRING(), SUBSTR(), MID()**: devolve uma substring de uma string.
- **TRIM()**: remove os espaços em branco do início e do final da string.

- **UNHEX()**: converte cada par de valores hexadecimais num carácter válido.
- **UPPER(), UCASE()**: coloca a string em maiúsculas.

A seguir, é proposto um exemplo do uso de várias das funções anteriores.




```
SELECT
ASCII('A'),
HEX('A'),
UPPER('Batata'),
LOWER('SARDINHA'),
REVERSE('SARDINHA'),
RTRIM(REPEAT('OK ', 3)),
SUBSTRING('Substring', 4),
SUBSTRING('Substring', 4, 6);
```

Por fim, serão apresentadas as funções meramente informativas que lhe podem interessar, como as que se encontram no website http://dev.mysql.com/doc/refman/5.1/en/informações-functions.html.

- **BENCHMARK()**: executa a expressão expr repetidamente count vezes. Pode ser usado para ver a rapidez com que o MySQL processa a expressão. O valor do resultado é sempre 0. O uso pretendido é de dentro do MySQL, que relata os tempos de execução da consulta.

  O tempo relatado é o tempo decorrido no cliente final, não o tempo de CPU no servidor. Recomenda-se executar o BENCHMARK() várias vezes e interpretar o resultado tendo em consideração a carga na máquina do servidor.

- **CHARSET()**: devolve o conjunto de caracteres do argumento.

- **COERCIBILITY()**: devolve a coerência do agrupamento do argumento, conforme mostrado na tabela a seguir:

Coercível	Significado	Exemplo
0	Agrupamento explícito	Valor com cláusula COLLATE.
1	Sem agrupamento	Concatenar strings com diferentes agrupamentos.
2	Agrupamento implícito	Valor de coluna.
3	Constante do sistema	Valor de retorno USER()
4	Coercível	String literal.
5	Ignorável	NULL ou uma expressão derivada de NULL.

- **COLLATION()**: devolve o conjunto de caracteres da string fornecida.

- **CONNECTION_ID()**: devolve o identificador da conexão.

- **CURRENT_USER(), CURRENT_USER**: devolve o nome do utilizador autenticado.

- **DATABASE(), SCHEMA()**: devolve o nome da base de dados atualmente em uso.

- **FOUND_ROWS()**: para uma consulta de seleção com a cláusula LIMIT, devolve o número de linhas que devolveria se a cláusula LIMIT não fosse usada.
- **LAST_INSERT_ID()**: devolve o valor da coluna do tipo AUTOINCREMENT para o último pedido de inserção.
- **ROW_COUNT()**: devolve o número de linhas atualizadas.
- **USER(), SESSION_USER(), SYSTEM_USER()**: devolve o nome do utilizador.
- **VERSION()**: devolve a versão do servidor MySQL.

Segue-se um exemplo do uso de algumas destas funcionalidades, apenas para fins informativos.

```
SELECT
VERSION() AS 'Versão do MySQL',
USER() AS 'Utilizador atual',
CONNECTION_ID() AS 'Identificador da conexão',
DATABASE() AS 'Nome da BD';
```

2. SELECT

Neste ponto irá estudar consultas de seleção, e para isso utilizará novamente a palavra **SELECT**. Desta vez, também irá aprender outras palavras com um significado especial que o acompanham para responder às perguntas dos utilizadores. Mais informações sobre as consultas selecionadas e a instrução SELECT podem ser encontradas na página web http://dev.mysql.com/doc/refman/5.1/en/select.html ou na ajuda da linha de comandos.

Nas consultas de seleção, a palavra **SELECT** é sempre acompanhada pela palavra **FROM**. A palavra FROM indica a tabela ou tabelas a serem consultadas e a palavra SELECT, a coluna ou colunas a serem exibidas. Poderá comprová-lo com o exemplo seguinte. É fundamental trabalhar com a base de dados adequada, neste caso a base de dados de futebol "USE futebol".

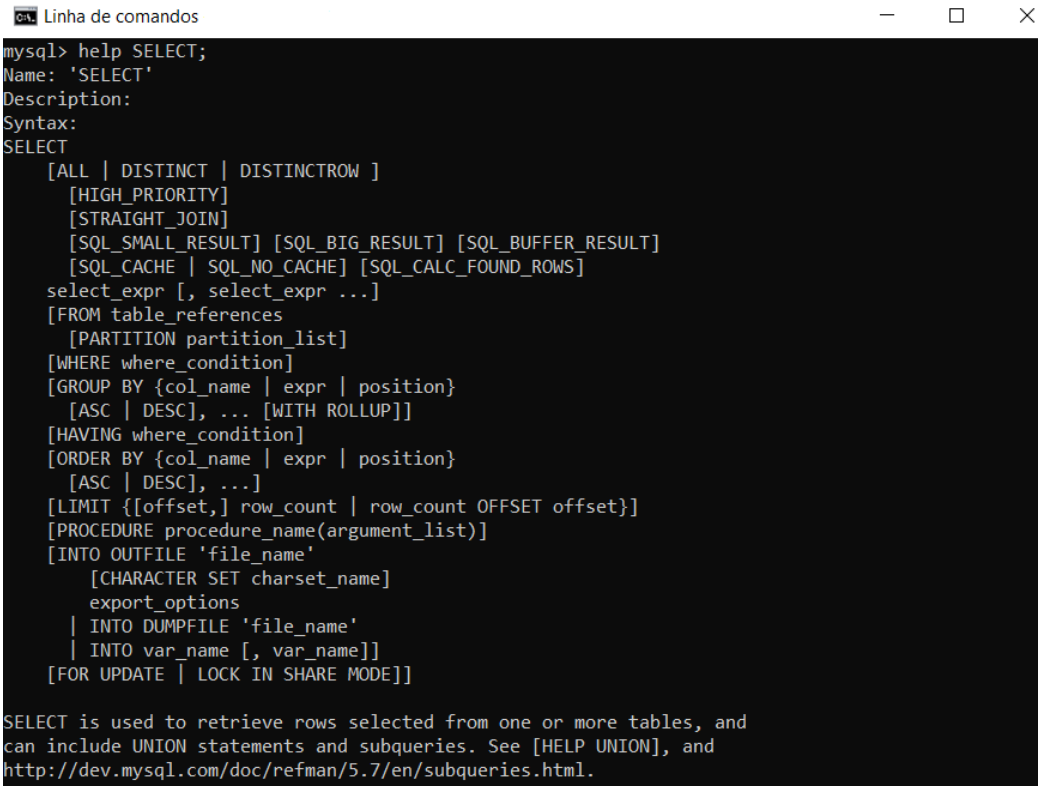
```
SELECT nomeEquipa FROM equipa;
```

A consulta anterior devolve a lista dos nomes das equipas que existem na base de dados. Como visto anteriormente, pode alterar o nome que será visto na janela das células do cabeçalho. Esta mudança é obtida a partir da palavra **AS**.

```
SELECT nomeEquipa AS 'Nome da equipa'FROM equipa;
```

Existem muitas outras ocasiões em que se pretende apenas rapidamente ver toda a tabela, não importando como as colunas são chamadas ou como são exibidas na janela. Nestes casos, pode utilizar um asterisco (*) na instrução SELECT, indicando assim que pretende mostrar todas as colunas da tabela.

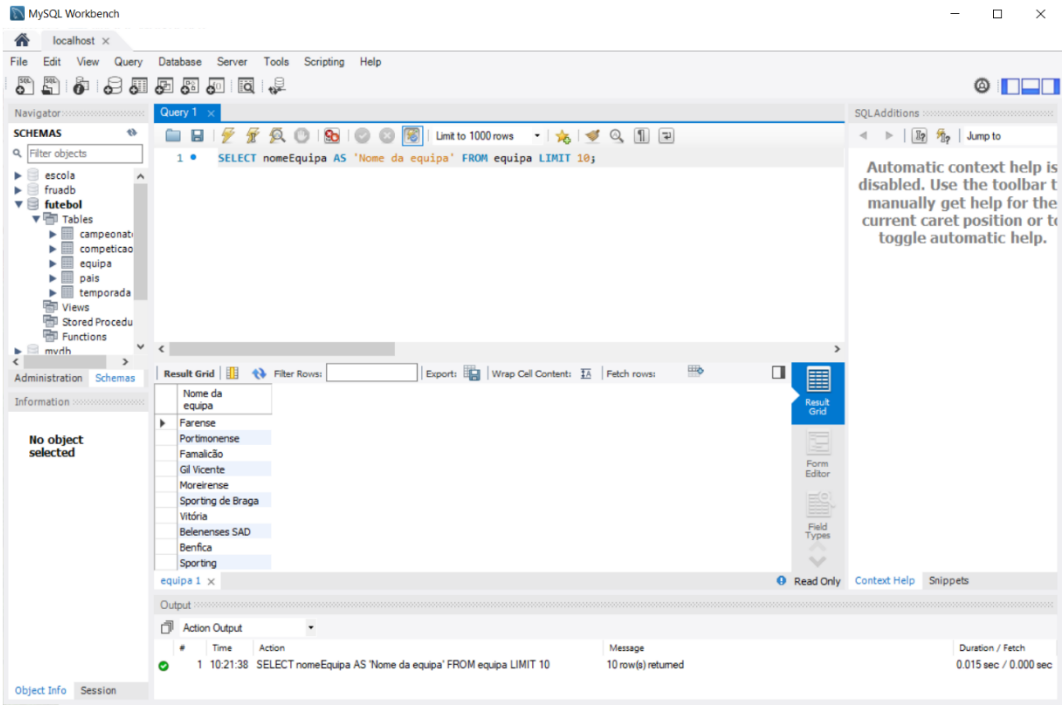
```
SELECT *FROM equipa;
```



Como pode ver na imagem, as consultas de seleção possuem muitas cláusulas e até agora foram utilizadas apenas duas delas. Aos poucos irá aprender as restantes. Deve notar que nem todas fazem parte do padrão SQL, mas, se estiver a usar o MySQL, poderá usá-las. Um exemplo de uma instrução que não faz parte da norma é a instrução **LIMIT**, que é mostrada a seguir.

```
SELECT nomeEquipa AS 'Nome da equipa'FROM equipa LIMIT 10;
```

Esta instrução, ao contrário das anteriores, irá devolver apenas o número de linhas indicado na palavra **LIMIT**. Ao contrário do que aconteceu com as instruções anteriores, que devolviam todas as linhas da tabela "equipa", desta vez apenas 10 linhas serão devolvidas, que podem ser quaisquer linhas da tabela, mas apenas 10. O resultado pode ser visto na imagem seguinte.



Importa destacar que, embora na frase anterior não tenha sido indicado de nenhuma forma que os resultados apareçam em ordem alfabética, estes surgem desse modo, porque está apenas a mostrar a coluna "nomeEquipa", uma coluna para a qual se definiu anteriormente que teria um índice único com classificação ascendente.

3. WHERE

Como foi visto até agora, é possível devolver todos os registos de uma tabela, porém muitas vezes são necessários resultados mais específicos. A forma de filtrar os resultados é por meio da instrução **WHERE**.

Por exemplo, ao trabalhar com a tabela "equipa", já viu que havia equipas com valores diferentes na coluna "paisID", embora o mais comum seja POR. O valor POR, como sabe, é o valor para Portugal, portanto, todas as equipas que não possuem esse identificador são equipas estrangeiras. A seguir foram preparadas duas listas: uma de todas as equipas portuguesas registadas na base de dados e outra de todas as equipas estrangeiras.

```
SELECT *FROM equipa WHERE paisID='POR';
```

equipaID	nomeEquipa	paisID
2	Farense	POR
3	Portimonense	POR
4	Famalicão	POR
5	Gil Vicente	POR
6	Moreirense	POR
7	Sporting de Braga	POR
8	Vitória	POR

equipaID	nomeEquipa	paisID
9	Belenenses SAD	POR
10	Benfica	POR
11	Sporting	POR
12	Marítimo	POR
13	Nacional	POR
14	Santa Clara	POR
15	Boavista	POR
16	Paços de Ferreira	POR
17	Porto	POR
18	Rio Ave	POR
19	Tondela	POR
20	Arouca	POR
21	Feirense	POR
22	Oliveirense	POR
23	Vizela	POR
24	Sporting da Covilhã	POR
25	Académica OAF	POR
26	Benfica B	POR
27	Casa Pia	POR
28	Estoril Praia	POR
29	Mafra	POR
30	Vilafranquense	POR
31	Leixões	POR
32	Penafiel	POR
33	Porto B	POR
34	Varzim	POR
35	Cova da Piedade	POR
36	Chaves	POR
37	Académico de Viseu	POR
77	Vitória de Guimarães	POR

equipaID	nomeEquipa	paisID
78	Olhanense	POR

Como pode verificar, para obter a lista de todas as equipas portuguesas, basta impor a condição de que a coluna “paisID” seja igual à string POR através do operador de igualdade (=). Caso queira mostrar as equipas estrangeiras presentes na base de dados, basta alterar a condição através do operador diferente de "<>", já que, se a equipa não é portuguesa, só pode ser estrangeira.

```
SELECT *FROM equipa WHERE paisID<>'POR';
```

equipaID	nomeEquipa	paisID
38	Real Madrid	ESP
39	Milan	ITA
40	Bayern de Munique	GER
41	Liverpool	ENG
42	Barcelona	ESP
43	Ajax	NED
44	Manchester United	ENG
45	Internazionale	ITA
46	Juventus	ITA
47	Nottingham Forest	ENG
48	Celtic	SCO
49	Hamburgo	GER
50	Steaua Bucareste	ROU
51	Olympique de Marseille	FRA
52	Borussia Dortmund	GER
53	Chelsea	ENG
54	Feyenoord	NED
55	Aston Villa	ENG
56	PSV Eindhoven	NED
57	Estrela Vermelha	SRB

equipaID	nomeEquipa	paisID
58	Atlético de Madrid	ESP
59	Stade de Reims	FRA
60	Valencia	ESP
61	Fiorentina	ITA
62	Eintracht Frankfurt	GER
63	Partizan	SRB
64	Panathinaikos	GRE
65	Leeds United	ENG
66	Saint Etienne	FRA
67	Borussia Mönchengladbach	GER
68	Brugge	BEL
69	Malmö	SWE
70	Roma	ITA
71	Sampdoria	ITA
72	Bayer Leverkusen	GER
73	Mónaco	FRA
74	Arsenal	ENG
75	Tottenham	ENG
76	Paris Saint-Germain	FRA
38	Real Madrid	ESP
39	Milan	ITA
40	Bayern de Munique	GER
41	Liverpool	ENG
42	Barcelona	ESP

Outro exemplo de filtragem é pelos operadores **LIKE** e **NOTLIKE**, que permitem avaliar se uma determinada coluna contém uma string específica. Estes dois operadores permitem o uso de caracteres para uma posição "_" ou várias posições "%". No caso de desejar pesquisar strings que tenham qualquer um dos daqueles caracteres, basta adicionar o carácter "\" no início. O operador **LIKE** exibe todos os resultados que correspondem à string especificada e o operador **NOT LIKE** exibe todos aqueles que não correspondem.

Segue-se um exemplo de todas as equipas que contêm a string **sport**, e outro em que são mostradas apenas as equipas que começam com essa sequência. Desta forma, poderá filtrar qualquer resultado por string inicial, string final ou uma string que pode ser encontrada em qualquer parte do texto. Desta forma, pode filtrar as equipas, apelidos e nomes de pessoas, entre outros. Por exemplo, se pretender saber o nome de todas as equipas que começam ou terminam com a letra "s".

```
SELECT * FROM equipa WHERE nomeEquipa LIKE '%sport%';
```

equipaID	nomeEquipa	paisID
7	Sporting de Braga	POR
11	Sporting	POR
24	Sporting da Covilhã	POR

```
SELECT * FROM equipa WHERE nomeEquipa LIKE 'sport%';
```

equipaID	nomeEquipa	paisID
7	Sporting de Braga	POR
11	Sporting	POR
24	Sporting da Covilhã	POR

```
SELECT * FROM equipa WHERE nomeEquipa LIKE 's%';
```

equipaID	nomeEquipa	paisID
7	Sporting de Braga	POR
11	Sporting	POR
14	Santa Clara	POR
24	Sporting da Covilhã	POR
50	Steaua Bucareste	ROU
59	Stade de Reims	FRA

equipaID	nomeEquipa	paisID
66	Saint Etienne	FRA
71	Sampdoria	ITA

```
SELECT * FROM equipa WHERE nomeEquipa LIKE '%s';
```

equipaID	nomeEquipa	paisID
31	Leixões	POR
36	Chaves	POR
46	Juventus	ITA
59	Stade de Reims	FRA
64	Panathinaikos	GRE
77	Vitória de Guimarães	POR

Outro operador interessante para utilizar com o MySQL é o BETWEEN, que permite verificar se um valor está num determinado intervalo. Por exemplo, no caso da tabela "temporada", pode ser necessário selecionar apenas as temporadas que estão entre 1983 e 1993. Segue-se um exemplo de como seria possível fazê-lo.

```
SELECT * FROM temporadaWHERE anoInicio BETWEEN 1983 AND 1993;
```

temporadaID	anoInicio	anoFim
92	1983	1984
93	1984	1985
94	1985	1986
95	1986	1987
96	1987	1988
97	1988	1989
98	1989	1990
99	1990	1991

temporadaID	anoInicio	anoFim
100	1991	1992
101	1992	1993
102	1993	1994

O operador **BETWEEN** não funciona apenas com números, também é possível utilizá-lo com letras e strings. Segue-se um exemplo em que todas as equipas existentes entre "r" e "ro" são mostradas como strings iniciais do nome.

```
SELECT * FROM equipa WHERE nomeEquipa BETWEEN 'r%' AND 'ro%';
```

equipaID	nomeEquipa	paisID
18	Rio Ave	POR
38	Real Madrid	ESP

Até agora, foi utilizada apenas uma condição para filtrar os resultados da pesquisa, embora seja possível utilizar várias condições de pesquisa através dos operadores condicionais **AND** e **OR**. No caso do operador **AND** é necessário que todas as condições sejam cumpridas ao mesmo tempo, enquanto no caso do operador OR basta que uma das condições seja cumprida, independentemente de qual for. No exemplo a seguir, será obtida a lista de equipas cujo identificador de país é POR e também as strings iniciais dos seus nomes que estão entre “r” e “ro”. O resultado, como se pode deduzir da tabela acima, só pode ser “Rio Ave”.

```
SELECT * FROM equipa WHERE (paisID='POR') AND (nomeEquipa BETWEEN
'r%' AND 'ro%');
```

No caso de optar pelo operador condicional OR, serão devolvidos todos os registos que cumpram uma ou outra condição, ou ambas ao mesmo tempo. Neste último caso, estarão presentes todas as equipas da tabela anterior, mais todas as equipas portuguesas presentes na base de dados.

```
SELECT * FROM equipa WHERE (paisID='POR') OR (nomeEquipa BETWEEN
'r%' AND 'ro%');
```

4. CONSULTAS QUE ENVOLVEM MAIS DO QUE UMA TABELA

Até agora trabalhou sempre com uma única tabela em cada caso, porém, em bases de dados relacionais, algumas tabelas estão relacionadas com outras. Neste ponto, serão aproveitadas as relações entre as tabelas para obter alguns resultados relevantes. Segue-se um exemplo para ver o que acontece se consultar as tabelas “equipa” e “pais” sem especificar nenhuma condição.

```
SELECT * FROM equipa, pais;
```

O resultado da execução da consulta acima são 16170 linhas devolvidas, mas sem qualquer significado. Especificamente, são reunidas cada uma das 77 equipas da base de dados com cada um dos 210 países. Se multiplicar as 77 equipas pelos 210 países, verifica-se que aparecem os 16170 resultados. Mas não era isso que se pretendia, pois cada equipa tem apenas um país. Por exemplo, é possível obter resultados relevantes da tabela das equipas e da tabela do país, como na consulta a seguir, na qual os dados relacionados são obtidos das duas tabelas e de todas as equipas. Neste caso, foi utilizado o valor "!=" no lugar de "<>" e obteve-se o mesmo resultado.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
e.paisID AS 'ID País',
p.paisID,
```

```
p.paisNome AS 'Nome do país'
FROM
equipa e,
pais p
WHERE
(e.paisID = p.paisID);
```

O resultado da execução da consulta anterior devolve as 77 equipas com o seu país correspondente, como pode ser visto na tabela a seguir. Embora não tenha sido necessário na consulta anterior, foi incluído "paisID" de ambas as tabelas para verificar visualmente se a condição de igualdade foi cumprida.

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
2	Farense	POR	POR	Portugal
3	Portimonense	POR	POR	Portugal
4	Famalicão	POR	POR	Portugal
5	Gil Vicente	POR	POR	Portugal
6	Moreirense	POR	POR	Portugal
7	Sporting de Braga	POR	POR	Portugal
8	Vitória	POR	POR	Portugal
9	Belenenses SAD	POR	POR	Portugal
10	Benfica	POR	POR	Portugal
11	Sporting	POR	POR	Portugal
12	Marítimo	POR	POR	Portugal
13	Nacional	POR	POR	Portugal
14	Santa Clara	POR	POR	Portugal
15	Boavista	POR	POR	Portugal
16	Paços de Ferreira	POR	POR	Portugal
17	Porto	POR	POR	Portugal
18	Rio Ave	POR	POR	Portugal
19	Tondela	POR	POR	Portugal
20	Arouca	POR	POR	Portugal
21	Feirense	POR	POR	Portugal

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
22	Oliveirense	POR	POR	Portugal
23	Vizela	POR	POR	Portugal
24	Sporting da Covilhã	POR	POR	Portugal
25	Académica OAF	POR	POR	Portugal
26	Benfica B	POR	POR	Portugal
27	Casa Pia	POR	POR	Portugal
28	Estoril Praia	POR	POR	Portugal
29	Mafra	POR	POR	Portugal
30	Vilafranquense	POR	POR	Portugal
31	Leixões	POR	POR	Portugal
32	Penafiel	POR	POR	Portugal
33	Porto B	POR	POR	Portugal
34	Varzim	POR	POR	Portugal
35	Cova da Piedade	POR	POR	Portugal
36	Chaves	POR	POR	Portugal
37	Académico de Viseu	POR	POR	Portugal
38	Real Madrid	ESP	ESP	Espanha
39	Milan	ITA	ITA	Itália
40	Bayern de Munique	GER	GER	Alemanha
41	Liverpool	ENG	ENG	Inglaterra
42	Barcelona	ESP	ESP	Espanha
43	Ajax	NED	NED	Países Baixos
44	Manchester United	ENG	ENG	Inglaterra
45	Internazionale	ITA	ITA	Itália
46	Juventus	ITA	ITA	Itália
47	Nottingham Forest	ENG	ENG	Inglaterra
48	Celtic	SCO	SCO	Escócia
49	Hamburgo	GER	GER	Alemanha
50	Steaua Bucareste	ROU	ROU	Romênia
51	Olympique de Marseille	FRA	FRA	França

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
52	Borussia Dortmund	GER	GER	Alemanha
53	Chelsea	ENG	ENG	Inglaterra
54	Feyenoord	NED	NED	Países Baixos
55	Aston Villa	ENG	ENG	Inglaterra
56	PSV Eindhoven	NED	NED	Países Baixos
57	Estrela Vermelha	SRB	SRB	Sérvia
58	Atlético de Madrid	ESP	ESP	Espanha
59	Stade de Reims	FRA	FRA	França
60	Valencia	ESP	ESP	Espanha
61	Fiorentina	ITA	ITA	Itália
62	Eintracht Frankfurt	GER	GER	Alemanha
63	Partizan	SRB	SRB	Sérvia
64	Panathinaikos	GRE	GRE	Grécia
65	Leeds United	ENG	ENG	Inglaterra
66	Saint Etienne	FRA	FRA	França
67	Borussia Mönchengladbach	GER	GER	Alemanha
68	Brugge	BEL	BEL	Bélgica
69	Malmö	SWE	SWE	Suécia
70	Roma	ITA	ITA	Itália
71	Sampdoria	ITA	ITA	Itália
72	Bayer Leverkusen	GER	GER	Alemanha
73	Mónaco	FRA	FRA	França
74	Arsenal	ENG	ENG	Inglaterra
75	Tottenham	ENG	ENG	Inglaterra
76	Paris Saint-Germain	FRA	FRA	França
77	Vitória de Guimarães	POR	POR	Portugal
78	Olhanense	POR	POR	Portugal

Além de filtrar que os dois "paisID" são iguais, também é possível filtrar, por exemplo, que as equipas são estrangeiras. Para fazer isso, adicione a condição **AND (p.nomePais != 'Portugal')** à consulta anterior. O resultado da execução desta consulta será uma tabela apenas com as equipas estrangeiras onde são adicionadas algumas colunas que indicam o nome do país. Os identificadores de país não devem ser incluídos na cláusula **SELECT**, se pretender vê-la como uma consulta mais definitiva.

Até agora não foi utilizada mais do que a cláusula **WHERE** para juntar resultados de várias tabelas, embora o MySQL ofereça uma possibilidade muito interessante chamada **JOIN**. Sobre a sua sintaxe e o seu uso pode consultar a página web http://dev.mysql.com/doc/refman/5.1/en/join.html. Por enquanto, basta saber que permitirá declarar as relações entre as tabelas na cláusula FROM, deixando a cláusula WHERE apenas para as verdadeiras condições de filtragem. Seguem-se várias consultas, utilizando a nova sintaxe, que irão devolver um resultado igual à tabela anterior.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
e.paisID AS 'ID País',
p.paisID,
p.paisNome AS 'Nome do país'
FROM
equipa e
JOIN pais p ON (e.paisID = p.paisID);
```

Se não for indicado o tipo de JOIN que se pretende fazer, por padrão, é realizada uma união interna INNERJOIN, mas é melhor indicá-lo explicitamente, pois existem mais tipos de JOIN e, assim, adquire a prática de especificar sempre o tipo, como mostra a consulta a seguir.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
e.paisID AS 'ID País',
```

```
p.paisID,
p.paisNome AS 'Nome do país'
FROM
equipa e
INNER JOIN pais p ON (e.paisID = p.paisID);
```

Quando ocorrer a situação de as colunas relacionadas das duas tabelas serem chamadas exatamente da mesma forma, é possível realizar a união com a cláusula **USING**. Quando este não for o caso, é possível utilizar apenas a palavra **ON**.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
e.paisID AS 'ID País',
p.paisID,
p.nomePais AS 'Nome do país'
FROM
equipa e
INNER JOIN pais p USING (paisID);
```

Nos casos anteriores foi utilizado o INNER JOIN, mas existem outras variantes muito interessantes, como LEFT e RIGHT. Voltando ao caso em que só interessava reunir as equipas estrangeiras, se realizar a consulta com INNERJOIN, serão devolvidas 39 linhas. Se escolher LEFT JOIN serão devolvidas novamente as mesmas 39, mas, se escolher RIGHT JOIN, irá obter um resultado surpreendente com 236 linhas.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipo',
e.paisID AS 'ID País',
p.paisID,
p.paisNome AS 'Nome do país'
```

```
FROM
equipa e
RIGHT JOIN pais p ON (e.paisID = p.paisID)
WHERE
p.paisNome != 'Portugal';
```

O resultado de realizar a consulta com RIGHT JOIN implica que todos os registos são retirados da tabela à direita ("pais"), que cumprem as condições WHERE. Neste caso, todos os países são devolvidos, exceto Portugal, haja ou não equipas desses mesmos países, e, nos casos em que haja várias equipas de um determinado país, é devolvido um registo para cada equipa. Como pode observar, existe um valor que se repete muito, o valor **NULL**, que indica vazio. Além disso, a igualdade imposta a "paisID" não foi cumprida, e isso aconteceu porque a recuperação de todos os registos à direita prevalece sobre o facto de serem iguais.

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	AFG	Afeganistão
NULL	NULL	NULL	RSA	África do Sul
NULL	NULL	NULL	ALB	Albânia
40	Bayern de Munique	GER	GER	Alemanha
49	Hamburgo	GER	GER	Alemanha
52	Borussia Dortmund	GER	GER	Alemanha
62	Eintracht Frankfurt	GER	GER	Alemanha
67	Borussia Mönchengladbach	GER	GER	Alemanha
72	Bayer Leverkusen	GER	GER	Alemanha
NULL	NULL	NULL	AND	Andorra
NULL	NULL	NULL	ANG	Angola
NULL	NULL	NULL	AIA	Anguilla
NULL	NULL	NULL	ATG	Antígua e Barbuda
NULL	NULL	NULL	KSA	Arábia Saudita
NULL	NULL	NULL	ALG	Argélia
NULL	NULL	NULL	ARG	Argentina

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	ARM	Arménia
NULL	NULL	NULL	ARU	Aruba
NULL	NULL	NULL	AUS	Austrália
NULL	NULL	NULL	AUT	Áustria
NULL	NULL	NULL	AZE	Azerbaijão
NULL	NULL	NULL	BAH	Bahamas
NULL	NULL	NULL	BHR	Bahrein
NULL	NULL	NULL	BAN	Bangladesh
NULL	NULL	NULL	BRB	Barbados
68	Brugge	BEL	BEL	Bélgica
NULL	NULL	NULL	BLZ	Belize
NULL	NULL	NULL	BEN	Benim
NULL	NULL	NULL	BER	Bermudas
NULL	NULL	NULL	BLR	Bielorrússia
NULL	NULL	NULL	BOL	Bolívia
NULL	NULL	NULL	BIH	Bósnia e Herzegovina
NULL	NULL	NULL	BOT	Botsuana
NULL	NULL	NULL	BRA	Brasil
NULL	NULL	NULL	BRU	Brunei
NULL	NULL	NULL	BUL	Bulgária
NULL	NULL	NULL	BFA	Burkina Faso
NULL	NULL	NULL	BDI	Burundi
NULL	NULL	NULL	BHU	Butão
NULL	NULL	NULL	CPV	Cabo Verde
NULL	NULL	NULL	CMR	Camarões
NULL	NULL	NULL	CAM	Camboja
NULL	NULL	NULL	CAN	Canadá
NULL	NULL	NULL	QAT	Catar
NULL	NULL	NULL	KAZ	Cazaquistão
NULL	NULL	NULL	CHA	Chade

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	CHI	Chile
NULL	NULL	NULL	CHN	China
NULL	NULL	NULL	CYP	Chipre
NULL	NULL	NULL	COL	Colômbia
NULL	NULL	NULL	COM	Comores
NULL	NULL	NULL	CGO	Congo
NULL	NULL	NULL	PRK	Coreia do Norte
NULL	NULL	NULL	KOR	Coreia do Sul
NULL	NULL	NULL	CIV	Costa de Marfim
NULL	NULL	NULL	CRC	Costa Rica
NULL	NULL	NULL	CRO	Croácia
NULL	NULL	NULL	CUB	Cuba
NULL	NULL	NULL	CUW	Curaçao
NULL	NULL	NULL	DEN	Dinamarca
NULL	NULL	NULL	DJI	Djibouti
NULL	NULL	NULL	DMA	Dominica
NULL	NULL	NULL	EGY	Egito
NULL	NULL	NULL	SLV	El Salvador
NULL	NULL	NULL	UAE	Emirados Árabes Unidos
NULL	NULL	NULL	ECU	Equador
NULL	NULL	NULL	ERI	Eritreia
48	Celtic	SCO	SCO	Escócia
NULL	NULL	NULL	SVK	Eslováquia
NULL	NULL	NULL	SVN	Eslovénia
38	Real Madrid	ESP	ESP	Espanha
42	Barcelona	ESP	ESP	Espanha
58	Atlético de Madrid	ESP	ESP	Espanha
60	Valencia	ESP	ESP	Espanha
NULL	NULL	NULL	ESW	Essuatíni
NULL	NULL	NULL	USA	Estados Unidos

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	EST	Estónia
NULL	NULL	NULL	ETH	Etiópia
NULL	NULL	NULL	FIJ	Fiji
NULL	NULL	NULL	PHI	Filipinas
NULL	NULL	NULL	FIN	Finlândia
51	Olympique de Marseille	FRA	FRA	França
59	Stade de Reims	FRA	FRA	França
66	Saint Etienne	FRA	FRA	França
73	Monaco	FRA	FRA	França
76	Paris Saint-Germain	FRA	FRA	França
NULL	NULL	NULL	GAB	Gabão
NULL	NULL	NULL	GAM	Gâmbia
NULL	NULL	NULL	GHA	Gana
NULL	NULL	NULL	GEP	Geórgia
NULL	NULL	NULL	GIB	Gibraltar
NULL	NULL	NULL	GRN	Granada
64	Panathinaikos	GRE	GRE	Grécia
NULL	NULL	NULL	GUM	Guam
NULL	NULL	NULL	GUA	Guatemala
NULL	NULL	NULL	GUY	Guiana
NULL	NULL	NULL	GUI	Guiné
NULL	NULL	NULL	GNB	Guiné Bissau
NULL	NULL	NULL	EQG	Guiné Equatorial
NULL	NULL	NULL	HAI	Haiti
NULL	NULL	NULL	HON	Honduras
NULL	NULL	NULL	HKG	Hong Kong
NULL	NULL	NULL	HUN	Hungria
NULL	NULL	NULL	YEM	Iémen
NULL	NULL	NULL	CAY	Ilhas Caimão
NULL	NULL	NULL	COK	Ilhas Cook

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	FRO	Ilhas Faroé
NULL	NULL	NULL	SOL	Ilhas Salomão
NULL	NULL	NULL	TCA	Ilhas Turks e Caicos
NULL	NULL	NULL	VIR	Ilhas Virgens Americanas
NULL	NULL	NULL	VGB	Ilhas Virgens Britânicas
NULL	NULL	NULL	IND	Índia
NULL	NULL	NULL	IDN	Indonésia
41	Liverpool	ENG	ENG	Inglaterra
44	Manchester United	ENG	ENG	Inglaterra
47	Nottingham Forest	ENG	ENG	Inglaterra
53	Chelsea	ENG	ENG	Inglaterra
55	Aston Villa	ENG	ENG	Inglaterra
65	Leeds United	ENG	ENG	Inglaterra
74	Arsenal	ENG	ENG	Inglaterra
75	Tottenham	ENG	ENG	Inglaterra
NULL	NULL	NULL	IRN	Irão
NULL	NULL	NULL	IRQ	Iraque
NULL	NULL	NULL	IRL	Irlanda
NULL	NULL	NULL	NIR	Irlanda do Norte
NULL	NULL	NULL	ISL	Islândia
NULL	NULL	NULL	ISR	Israel
39	Milan	ITA	ITA	Itália
45	Internazionale	ITA	ITA	Itália
46	Juventus	ITA	ITA	Itália
61	Fiorentina	ITA	ITA	Itália
70	Roma	ITA	ITA	Itália
71	Sampdoria	ITA	ITA	Itália
NULL	NULL	NULL	JAM	Jamaica
NULL	NULL	NULL	JPN	Japão
NULL	NULL	NULL	JOR	Jordânia

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	KVX	Kosovo
NULL	NULL	NULL	KUW	Kuwait
NULL	NULL	NULL	LAO	Laos
NULL	NULL	NULL	LES	Lesoto
NULL	NULL	NULL	LVA	Letónia
NULL	NULL	NULL	LIB	Líbano
NULL	NULL	NULL	LBR	Libéria
NULL	NULL	NULL	LBY	Líbia
NULL	NULL	NULL	LIE	Liechtenstein
NULL	NULL	NULL	LTU	Lituânia
NULL	NULL	NULL	LUX	Luxemburgo
NULL	NULL	NULL	MAC	Macau
NULL	NULL	NULL	MKD	Macedónia do Norte
NULL	NULL	NULL	MAD	Madagáscar
NULL	NULL	NULL	MAS	Malásia
NULL	NULL	NULL	MWI	Malawi
NULL	NULL	NULL	MDV	Maldivas
NULL	NULL	NULL	MLI	Mali
NULL	NULL	NULL	MLT	Malta
NULL	NULL	NULL	MAR	Marrocos
NULL	NULL	NULL	MRI	Maurícia
NULL	NULL	NULL	MTN	Mauritânia
NULL	NULL	NULL	MEX	México
NULL	NULL	NULL	MOZ	Moçambique
NULL	NULL	NULL	MDA	Moldávia
NULL	NULL	NULL	MNG	Mongólia
NULL	NULL	NULL	MNE	Montenegro
NULL	NULL	NULL	MSR	Montserrat
NULL	NULL	NULL	MYA	Myanmar
NULL	NULL	NULL	NAM	Namíbia

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	NEP	Nepal
NULL	NULL	NULL	NCA	Nicarágua
NULL	NULL	NULL	NIG	Níger
NULL	NULL	NULL	NGA	Nigéria
NULL	NULL	NULL	NOR	Noruega
NULL	NULL	NULL	NCL	Nova Caledónia
NULL	NULL	NULL	NZL	Nova Zelândia
NULL	NULL	NULL	OMA	Omã
NULL	NULL	NULL	WAL	País de Gales
43	Ajax	NED	NED	Países Baixos
54	Feyenoord	NED	NED	Países Baixos
56	PSV Eindhoven	NED	NED	Países Baixos
NULL	NULL	NULL	PLE	Palestina
NULL	NULL	NULL	PAN	Panamá
NULL	NULL	NULL	PNG	Papua-Nova Guiné
NULL	NULL	NULL	PAK	Paquistão
NULL	NULL	NULL	PAR	Paraguai
NULL	NULL	NULL	PER	Peru
NULL	NULL	NULL	POL	Polónia
NULL	NULL	NULL	PUR	Porto Rico
NULL	NULL	NULL	KEN	Quénia
NULL	NULL	NULL	KGZ	Quirguistão
NULL	NULL	NULL	COD	RD Congo
NULL	NULL	NULL	CTA	República Centro-Africana
NULL	NULL	NULL	DOM	República Dominicana
50	Steaua Bucareste	ROU	ROU	Roménia
NULL	NULL	NULL	RWA	Ruanda
NULL	NULL	NULL	RUS	Rússia
NULL	NULL	NULL	SAM	Samoa

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	ASA	Samoa Americana
NULL	NULL	NULL	SMR	San Marino
NULL	NULL	NULL	LCA	Santa Lúcia
NULL	NULL	NULL	SKN	São Cristóvão e Névis
NULL	NULL	NULL	STP	São Tomé e Príncipe
NULL	NULL	NULL	VIN	São Vicente e Granadinas
NULL	NULL	NULL	SEN	Senegal
NULL	NULL	NULL	SLW	Serra Leoa
57	Estrela Vermelha	SRB	SRB	Sérvia
63	Partizan	SRB	SRB	Sérvia
NULL	NULL	NULL	SEY	Seychelles
NULL	NULL	NULL	SIN	Singapura
NULL	NULL	NULL	SYR	Síria
NULL	NULL	NULL	SOM	Somália
NULL	NULL	NULL	SRI	Sri Lanka
NULL	NULL	NULL	SDN	Sudão
NULL	NULL	NULL	SSD	Sudão do Sul
69	Malmö	SWE	SWE	Suécia
NULL	NULL	NULL	SUI	Suíça
NULL	NULL	NULL	SUR	Suriname
NULL	NULL	NULL	THA	Tailândia
NULL	NULL	NULL	TPE	Taipé Chinês
NULL	NULL	NULL	TAH	Taiti
NULL	NULL	NULL	TJK	Tajiquistão
NULL	NULL	NULL	TAN	Tanzânia
NULL	NULL	NULL	CZE	Chéquia
NULL	NULL	NULL	TLS	Timor-Leste
NULL	NULL	NULL	TOG	Togo
NULL	NULL	NULL	TGA	Trinidad e Tobago

ID Equipa	Nome da equipa	ID País	paisID	Nome do país
NULL	NULL	NULL	TUN	Tunísia
NULL	NULL	NULL	TKM	Turquemenistão
NULL	NULL	NULL	TUR	Turquia
NULL	NULL	NULL	UKR	Ucrânia
NULL	NULL	NULL	UGA	Uganda
NULL	NULL	NULL	URU	Uruguai
NULL	NULL	NULL	UZB	Uzbequistão
NULL	NULL	NULL	VAN	Vanuatu
NULL	NULL	NULL	VEN	Venezuela
NULL	NULL	NULL	VIE	Vietname
NULL	NULL	NULL	ZAM	Zâmbia
NULL	NULL	NULL	ZIM	Zimbabué

O valor NULL é, normalmente, um valor problemático, enigmático e um inimigo que se deve combater. Para evitar que os valores nulos sejam devolvidos é utilizada a cláusula WHERE, em que nenhum registo com valor nulo é devolvido na coluna "paisID"; para isso, utiliza-se **IS NOT NULL**. A execução da próxima query irá devolver os 39 resultados anteriores, ou seja, sem os valores NULL.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
e.paisID AS 'ID País',
p.paisID,
p.nomePais AS 'Nome do país'
FROM
equipa e
RIGHT JOIN pais p ON (e.paisID = p.paisID)
WHERE
(p.nomePais != 'Portugal')
AND (e.paisID IS NOT NULL);
```

5. ORDER BY

Até agora, foram sempre devolvidos resultados sem nenhuma ordem, tal como o servidor os forneceu, embora seja possível ordená-los por ordem alfabética ou numérica, crescente ou decrescente, de uma determinada coluna ou colunas. Para isso, é utilizada a cláusula **ORDER BY** das consultas de seleção (SELECT). Como exemplo, as duas consultas a seguir mostram os 5 primeiros países e os 5 últimos da grande lista de países por ordem alfabética. O resultado é restrito a 5 registos por uma questão de gestão do espaço, mas a lista completa pode ser devolvida sem problemas.

```
SELECT * FROM paisORDER BY paisNome ASCLIMIT 5;
```

paisID	paisNome
AFG	Afeganistão
RSA	África do Sul
ALB	Albânia
GER	Alemanha
AND	Andorra

```
SELECT * FROM paisORDER BY paisNomeDESCLIMIT 5;
```

paisID	paisNome
ZIM	Zimbabué
ZAM	Zâmbia
VIE	Vietname
VEN	Venezuela
VAN	Vanuatu

Além dos valores da string, também pode ordenar valores numéricos ou datas com o **ASC** (crescente) e o **DESC** (decrescente). No caso de a ordem de classificação não ser declarada explicitamente, presume-se que seja **ASC**. Além disso, os resultados podem ser ordenados sob vários critérios, e até mesmo alterando a ordem dos critérios, conforme mostrado nas consultas a seguir.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
p.paisNome AS 'Nome do país'
FROM
equipa e
INNER JOIN pais p ON (e.paisID = p.paisID)
WHERE
(e.nomeEquipa LIKE '%sport%')
ORDER BY
e.nomeEquipa ASC,
p.paisNomeASC;
```

Em ambas as consultas, são devolvidas as equipas que contêm a string "sport". A primeira é ordenada primeiro por equipa e depois por país, enquanto a segunda é ordenada primeiro por país e depois por equipa.

```
SELECT
e.equipaID AS 'ID Equipa',
e.nomeEquipa AS 'Nome da equipa',
p.paisNome AS 'Nome do país'
```

```
FROM
equipa e
INNER JOIN pais p ON (e.paisID = p.paisID)
WHERE
(e.nomeEquipa LIKE '%sport%')
ORDER BY
p.paisNomeASC,
e.nomeEquipa ASC;
```

Após estes exemplos simples, está pronto para resultados muito mais interessantes e complexos que abrangem muito mais tabelas; basta refinar um pouco uma consulta que foi usada nos pontos anteriores. Neste caso pretende-se recuperar a lista de campeões e segundos classificados de todas as competições que se iniciam com a palavra "Primeira", ordenada por época, e caso se repitam temporadas, também ordenada por equipa campeã. Como pode ser visto, todas as junções entre as tabelas foram feitas com **INNER JOIN**.

```
SELECT
CONCAT(CAST(temp.anoInicio AS CHAR(4)), ' - ' ,
CAST(temp.anoFim AS CHAR(4))) AS Temporada,
comp.competicaoNome AS Competição,
eq1.nomeEquipa AS Campeão,
eq2.nomeEquipa AS SegundoClassificado
FROM
campeonato camp
INNER JOIN competicao comp ON
(camp.competicaoID = comp.competicaoID)
INNER JOIN temporada temp ON
(camp.temporadaID = temp.temporadaID)
INNER JOIN equipa eq1 ON
(camp.campeao = eq1.equipaID)
LEFT JOIN equipa eq2 ON
(camp.subcampeao = eq2.equipaID)
WHERE
```

```
comp.competicaoNome LIKE 'Primeira%'
ORDER BY
Temporada,
Campeão;
```

Temporada	Competição	Campeão	Segundo classificado
1938 - 1939	Primeira Liga	Académica OAF	Benfica
1939 - 1940	Primeira Liga	Benfica	Belenenses SAD
1940 - 1941	Primeira Liga	Sporting	Belenenses SAD
1941 - 1942	Primeira Liga	Belenenses SAD	Vitória de Guimarães
1942 - 1943	Primeira Liga	Benfica	Vitória
1943 - 1944	Primeira Liga	Benfica	Estoril Praia
1944 - 1945	Primeira Liga	Sporting	Olhanense
1945 - 1946	Primeira Liga	Sporting	Académico de Viseu
1947 - 1948	Primeira Liga	Sporting	Belenenses SAD
1948 - 1949	Primeira Liga	Benfica	Académico de Viseu
1950 - 1951	Primeira Liga	Benfica	Académica OAF
1951 - 1952	Primeira Liga	Benfica	Sporting
1952 - 1953	Primeira Liga	Benfica	Porto
1953 - 1954	Primeira Liga	Sporting	Vitória
1954 - 1955	Primeira Liga	Benfica	Sporting
1955 - 1956	Primeira Liga	Porto	Tondela
1956 - 1957	Primeira Liga	Benfica	Sporting da Covilhã
1957 - 1958	Primeira Liga	Porto	Benfica
1958 - 1959	Primeira Liga	Benfica	Porto
1959 - 1960	Primeira Liga	Belenenses SAD	Sporting
1960 - 1961	Primeira Liga	Leixões	Porto
1961 - 1962	Primeira Liga	Benfica	Vitória
1962 - 1963	Primeira Liga	Sporting	Vitória de Guimarães
1963 - 1964	Primeira Liga	Benfica	Porto
1964 - 1965	Primeira Liga	Vitória	Benfica

Temporada	Competição	Campeão	Segundo classificado
1965 - 1966	Primeira Liga	Sporting de Braga	Vitória
1966 - 1967	Primeira Liga	Vitória	Académica OAF
1967 - 1968	Primeira Liga	Porto	Vitória
1968 - 1969	Primeira Liga	Benfica	Académica OAF
1969 - 1970	Primeira Liga	Benfica	Sporting
1970 - 1971	Primeira Liga	Sporting	Benfica
1971 - 1972	Primeira Liga	Benfica	Sporting
1972 - 1973	Primeira Liga	Sporting	Vitória
1973 - 1974	Primeira Liga	Sporting	Benfica
1974 - 1975	Primeira Liga	Boavista	Benfica
1975 - 1976	Primeira Liga	Boavista	Vitória de Guimarães
1976 - 1977	Primeira Liga	Porto	Sporting de Braga
1977 - 1978	Primeira Liga	Sporting	Porto
1978 - 1979	Primeira Liga	Boavista	Sporting
1979 - 1980	Primeira Liga	Benfica	Porto
1980 - 1981	Primeira Liga	Benfica	Porto
1981 - 1982	Primeira Liga	Sporting	Sporting de Braga
1982 - 1983	Primeira Liga	Benfica	Porto
1983 - 1984	Primeira Liga	Porto	Rio Ave
1984 - 1985	Primeira Liga	Benfica	Porto
1985 - 1986	Primeira Liga	Benfica	Belenenses SAD
1986 - 1987	Primeira Liga	Benfica	Sporting
1987 - 1988	Primeira Liga	Porto	Vitória de Guimarães
1988 - 1989	Primeira Liga	Belenenses SAD	Benfica
1989 - 1990	Primeira Liga	Arouca	Farense
1990 - 1991	Primeira Liga	Porto	Benfica B
1991 - 1992	Primeira Liga	Boavista	Porto
1992 - 1993	Primeira Liga	Benfica	Boavista
1993 - 1994	Primeira Liga	Porto	Sporting
1994 - 1995	Primeira Liga	Sporting	Marítimo

Temporada	Competição	Campeão	Segundo classificado
1995 - 1996	Primeira Liga	Benfica	Sporting
1996 - 1997	Primeira Liga	Boavista	Benfica
1997 - 1998	Primeira Liga	Porto	Sporting de Braga
1998 - 1999	Primeira Liga	Benfica B	Casa Pia
1999 - 2000	Primeira Liga	Porto	Sporting
2000 - 2001	Primeira Liga	Porto	Marítimo
2001 - 2002	Primeira Liga	Sporting	Leixões
2002 - 2003	Primeira Liga	Porto	Santa Clara
2003 - 2004	Primeira Liga	Benfica	Porto
2004 - 2005	Primeira Liga	Vitória	Benfica
2005 - 2006	Primeira Liga	Porto	Vitória
2006 - 2007	Primeira Liga	Sporting	Belenenses SAD
2007 - 2008	Primeira Liga	Sporting	Porto
2008 - 2009	Primeira Liga	Porto	Paços de Ferreira
2009 - 2010	Primeira Liga	Porto	Chaves
2010 - 2011	Primeira Liga	Porto	Vitória de Guimarães
2011 - 2012	Primeira Liga	Académica OAF	Sporting
2012 - 2013	Primeira Liga	Vitória de Guimarães	Benfica
2013 - 2014	Primeira Liga	Benfica	Rio Ave
2014 - 2015	Primeira Liga	Sporting	Sporting de Braga
2015 - 2016	Primeira Liga	Sporting de Braga	Porto
2016 - 2017	Primeira Liga	Benfica	Vitória de Guimarães
2017 - 2018	Primeira Liga	Rio Ave	Sporting
2018 - 2019	Primeira Liga	Sporting	Porto
2019 - 2020	Primeira Liga	Porto	Benfica
1938 - 1939	Primeira Liga	Académica OAF	Benfica
1939 - 1940	Primeira Liga	Benfica	Belenenses SAD
1940 - 1941	Primeira Liga	Sporting	Belenenses SAD
1941 - 1942	Primeira Liga	Belenenses SAD	Vitória de Guimarães

Temporada	Competição	Campeão	Segundo classificado
1942 - 1943	Primeira Liga	Benfica	Vitória
1943 - 1944	Primeira Liga	Benfica	Estoril Praia
1944 - 1945	Primeira Liga	Sporting	Olhanense
1945 - 1946	Primeira Liga	Sporting	Académico de Viseu
1947 - 1948	Primeira Liga	Sporting	Belenenses SAD
1948 - 1949	Primeira Liga	Benfica	Académico de Viseu
1950 - 1951	Primeira Liga	Benfica	Académica OAF
1951 - 1952	Primeira Liga	Benfica	Sporting
1952 - 1953	Primeira Liga	Benfica	Porto
1953 - 1954	Primeira Liga	Sporting	Vitória
1954 - 1955	Primeira Liga	Benfica	Sporting
1955 - 1956	Primeira Liga	Porto	Tondela
1956 - 1957	Primeira Liga	Benfica	Sporting da Covilhã
1957 - 1958	Primeira Liga	Porto	Benfica
1958 - 1959	Primeira Liga	Benfica	Porto
1959 - 1960	Primeira Liga	Belenenses SAD	Sporting
1960 - 1961	Primeira Liga	Leixões	Porto
1961 - 1962	Primeira Liga	Benfica	Vitória
1962 - 1963	Primeira Liga	Sporting	Vitória de Guimarães
1963 - 1964	Primeira Liga	Benfica	Porto
1964 - 1965	Primeira Liga	Vitória	Benfica
1965 - 1966	Primeira Liga	Sporting de Braga	Vitória
1966 - 1967	Primeira Liga	Vitória	Académica OAF
1967 - 1968	Primeira Liga	Porto	Vitória
1968 - 1969	Primeira Liga	Benfica	Académica OAF

Podem ser realizadas muitas consultas diferentes sob alguns dos critérios de ordenação de tabela que abordámos, condições de filtragem, junções entre tabelas, uso de funções de conversão, etc. Todas estas, por mais complexas que sejam, não serão muito mais complicadas do que as que acabou de experimentar. Precisará apenas de tempo e paciência para pensar sobre elas e refiná-las o mais possível.

6. FUNÇÕES AGREGADAS

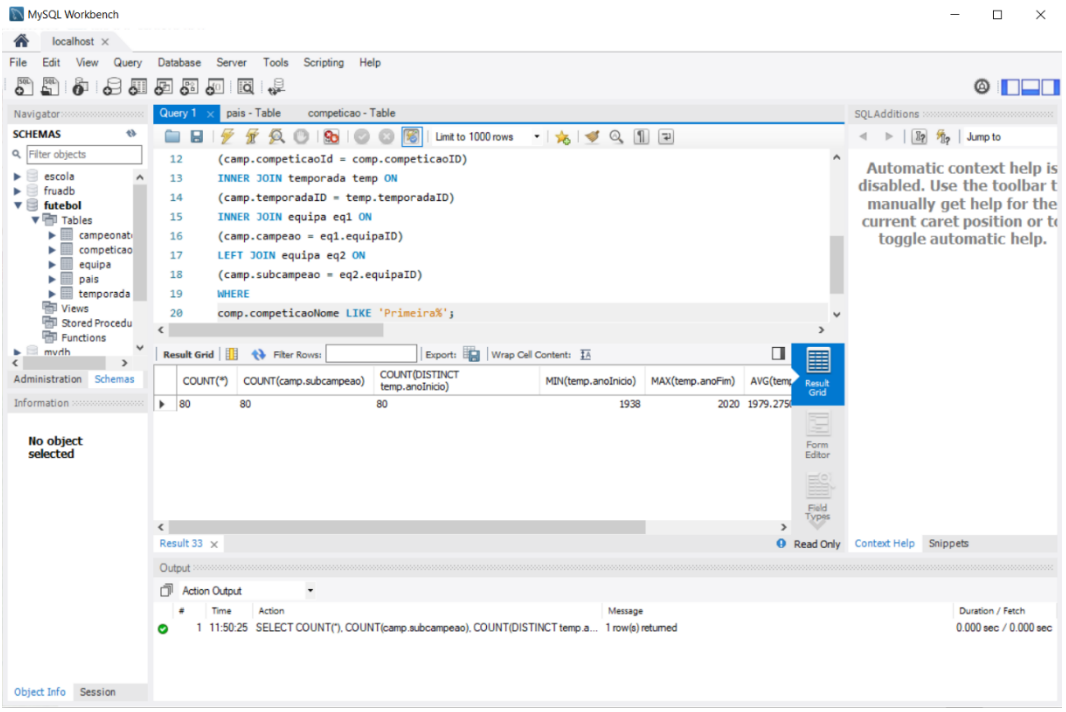
As funções de agregação permitem aplicar um cálculo a um conjunto de valores do resultado de uma consulta. Pode ver a lista de todas as funções em http://dev.mysql.com/doc/refman/5.1/en/group-by-functions.html, mas as mais relevantes são as seguintes:

- **AVG()**: devolve o valor médio de todos os registos.
- **COUNT()**: conta o número de registos devolvidos. Se usado com um asterisco, os valores nulos também são contados.
- **COUNT(DISTINT)**: o mesmo que acima, mas não conta os valores repetidos.
- **MAX()**: devolve o valor máximo de todos os devolvidos.
- **MIN()**: devolve o valor mínimo de todos os devolvidos.
- **SUM()**: devolve a soma.

Aqui está um exemplo bastante completo onde as principais funções de agregação vistas acima são utilizadas. Também mostra a diferença entre usar o valor***** ou outro com a função **COUNT**, e a possibilidade de usar o valor **DISTINCT** numa determinada coluna para indicar que valores repetidos não são contados.

```
SELECT
COUNT(*),
COUNT(camp.subcampeao),
COUNT(DISTINCT temp.anoInicio),
```

```
MIN(temp.anoInicio),
MAX(temp.anoFim),
AVG(temp.anoInicio),
SUM(temp.anoInicio)
FROM
campeonato camp
INNER JOIN competicao comp ON
(camp.competicaoId = comp.competicaoID)
INNER JOIN temporada temp ON
(camp.temporadaID = temp.temporadaID)
INNER JOIN equipa eq1 ON
(camp.campeao = eq1.equipaID)
LEFT JOIN equipa eq2 ON
(camp.subcampeao = eq2.equipaID)
WHERE
comp.competicaoNome LIKE 'Primeira%';
```



7. AGRUPAMENTO DE LINHAS

Um bom exemplo da importância de agrupar os resultados é encontrado nos resultados desportivos; muitas vezes, os comentadores falam sobre o facto de uma equipa ter mais títulos do que outra. A melhor forma de conhecer esses dados é agrupar os resultados utilizando a cláusula **GROUP BY**, que permite especificar sob que critérios os resultados são agrupados. Por exemplo, se pretender conhecer as equipas vencedoras das competições cujo nome comece com "Taça" e saber os títulos obtidos por cada uma delas, o mais fácil é agrupar pelo nome da equipa, como se vê no exemplo a seguir.

```
SELECT
eq1.nomeEquipa AS 'Equipavencedora',
COUNT(*) AS 'Troféus'
FROM
campeonato camp
INNER JOIN competicao comp ON
(camp.competicaoID = comp.competicaoID)
INNER JOIN temporada temp ON
(camp.temporadaID = temp.temporadaID)
INNER JOIN equipa eq1 ON
(camp.campeao = eq1.equipaID)
LEFT JOIN equipa eq2 ON
(camp.subcampeao = eq2.equipaID)
WHERE
```

```
comp.competicaoNome LIKE 'Primeira%'
GROUP BY
eq1.nomeEquipa
ORDER BY
Troféus DESC,
eq1.nomeEquipa ASC;
```

Equipa vencedora	Troféus
Benfica	26
Porto	17
Sporting	17
Boavista	5
Belenenses SAD	3
Vitória	3
Académica OAF	2
Sporting de Braga	2
Arouca	1
Estrela da Amadora	1
Leixões	1
Rio Ave	1
Vitória de Guimarães	1

A consulta anterior devolve todas as equipas vencedoras de um título cujo nome comece com "Primeira"; caso as equipas tenham o mesmo número de títulos, os dados são devolvidos ordenados pelo nome da equipa. Se pretender filtrar para que, por exemplo, apenas equipas com mais do que cinco títulos sejam considerados, poderá fazê-lo através da palavra **HAVING**. Esta permite que as condições sejam incluídas de forma semelhante à palavra WHERE, com a diferença de que HAVING só pode ser utilizada com funções e atributos agregados incluídos na cláusula **GROUP BY**. A consulta que mostra as equipas com mais de cinco títulos é demonstrada a seguir.

```
SELECT
```

```
eq1.nomeEquipa AS 'Equipa vencedora',
COUNT(*) AS 'Troféus'
FROM
campeonato camp
INNER JOIN competicao comp ON
(camp.competicaoID = comp.competicaoID)
INNER JOIN temporada temp ON
(camp.temporadaID = temp.temporadaID)
INNER JOIN equipa eq1 ON
(camp.campeao = eq1.equipaID)
LEFT JOIN equipa eq2 ON
(camp.subcampeao = eq2.equipaID)
WHERE
comp.competicaoNomeLIKE 'Primeira%'
GROUP BY
eq1.nomeEquipa
HAVING
COUNT(*) > 5
ORDER BY
Troféus DESC,
eq1.nomeEquipa ASC;
```

Por último, se pretender saber o nome de todas as equipas que alguma vez disputaram uma final de uma competição cujo nome comece com "Primeira", poderá fazê-lo modificando ligeiramente as consultas anteriores.

```
SELECT
eq2.nomeEquipa AS 'Segundo Classificado',
COUNT(eq2.nomeEquipa) AS
'Número de finais perdidas'
FROM
campeonato camp
INNER JOIN competicao comp ON
```

```
(camp.competicaoID = comp.competicaoID)
INNER JOIN temporada temp ON
(camp.temporadaID = temp.temporadaID)
INNER JOIN equipa eq1 ON
(camp.campeao = eq1.equipaID)
INNER JOIN equipa eq2 ON
(camp.subcampeao = eq2.equipaID)
WHERE
comp.competicaoNomeLIKE 'Primeira%'
GROUP BY
eq2.nomeEquipa
ORDER BY
COUNT(eq2.nomeEquipa) DESC,
eq2.nomeEquipa ASC;
```

Segundo classificado	Número de finais perdidas
Porto	14
Sporting	12
Benfica	11
Vitória	7
Vitória de Guimarães	6
Belenenses SAD	5
Sporting de Braga	4
Académica OAF	3
Académico de Viseu	2
Marítimo	2
Rio Ave	2
Estrela da Amadora	1
Boavista	1
Casa Pia	1
Chaves	1

Segundo classificado	Número de finais perdidas
Estoril Praia	1
Farense	1
Leixões	1
Olhanense	1
Paços de Ferreira	1
Santa Clara	1
Sporting da Covilhã	1
Tondela	1

CONCLUSÃO

Nesta unidade didática, trabalhou com a linha de comandos e com o MySQL Workbench, com o intuito de recuperar informações valiosas das tabelas da base de dados dedicada ao futebol. Como deve ter notado, os dados isolados têm pouco valor, mas se estiverem relacionados com os dados de outras tabelas, tornam-se muito importantes.

Assim, pôde finalmente recolher os frutos do trabalho das unidades anteriores, e certamente valeu a pena. Descobriu um novo mundo cheio de possibilidades e funcionalidades para testar e descobrir, o que poderá fazer todos os dias.

AUTOAVALIAÇÃO

1. **Qual é o carácter que permite especificar um número variável de caracteres?**

   a) O underscore: _.

   b) O cardinal: #.

   c) A barra: /.

   d) A percentagem: %.

2. **Qual das seguintes funções se aplica a valores numéricos?**

   a) CONCAT().

   b) SIN().

   c) ASCII().

   d) CURDATE().

3. **Qual é o valor que a função SELECT NOW(); devolve?**

   a) A data e a hora atuais.

   b) A data atual.

   c) A hora atual.

   d) Nenhum, porque dá um erro.

4. Qual das funções a seguir não tem a mesma funcionalidade que a função NOW()?

a) LOCALTIME().

b) LOCALTIMESTAMP().

c) CURDATE().

d) CURRENT_TIMESTAMP().

5. Qual das seguintes afirmações é verdadeira?

a) A palavra FROM permite especificar as tabelas a serem consultadas.

b) A palavra FROM permite especificar as colunas a serem consultadas.

c) A palavra WHERE é obrigatória.

d) A palavra WHERE permite agrupar resultados.

6. Para que serve o operador BETWEEN?

a) Para ignorar os valores nulos.

b) Para verificar se um determinado valor está dentro de uma determinada faixa de valores.

c) Para contar o número de registos devolvidos.

d) O operador BETWEEN não pode ser usado com MYSQL.

7. Qual das seguintes afirmações é verdadeira?

a) A cláusula GROUP BY é obrigatória.

b) A cláusula HAVING pode ser usada sem qualquer limitação.

c) A cláusula ORDER BY não pode ser usada com vários critérios de pedido ao mesmo tempo.

d) A palavra LIMIT pode ser usada com MySQL, mas não faz parte do padrão SQL.

8. Qual das seguintes expressões significa "diferente de"?

a) &%=.

b) =.

c) !=.

d) /=.

9. Qual é o padrão de pesquisa mais correto se pretender listar as equipas que contêm a palavra "Clube" algures no nome?

a) %Clube%.

b) Clube%.

c) _Clube_.

d) Clube_.

10. Qual é o padrão de pesquisa mais correto se pretender listar as equipas que começam com a palavra “Clube”?

a) %Club%.

b) Club%.

c) _Club_.

d) Club_.

SOLUÇÕES

1.	d	**2.**	b	**3.**	a	**4.**	c	**5.**	a
6.	b	**7.**	d	**8.**	c	**9.**	a	**10.**	b

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para expandir os conhecimentos adquiridos nesta unidade, recomenda-se que visite as seguintes páginas web:

- https://www.w3schools.com/sql/sql_syntax.asp
- https://www.w3schools.com/sql/sql_select.asp
- https://www.w3schools.com/sql/sql_where.asp
- https://www.w3schools.com/sql/sql_and_or.asp
- https://www.w3schools.com/sql/sql_orderby.asp

BIBLIOGRAFIA

- Beaulieu, A. (2006), Aprende SQL. Madrid: Anaya Multimedia.
- Gutiérrez Gallardo, J. D. (2009), MySQL 5.1. Madrid: Anaya Multimedia.
- Oracle (2021), "Reference Manual". Disponível em: http://dev.mysql.com/doc/refman/5.1/en/.

  Consultado a 16 de abril de 2021.