GUIA DE LABORATÓRIO 2.1
CONSULTAS SIMPLES

OBJECTIVOS

- Utilizar o comando `SELECT` para definir consultas com critérios de pesquisa
- Conhecer os vários operadores para definir esses critérios de pesquisas
- Manipular as colunas que resultam das consultas
- Definir consultas reutilizáveis através do comando `CREATE VIEW`

INSTRUÇÕES

1a Parte – Introdução ao Comando SELECT

1. Inicie o HeidiSQL da forma habitual.

2. Verifique na tabela `Curso` qual o `ID` do curso de '`Contabilidade`' (para todos os efeitos, vamos supor que os códigos dos três cursos na tabela `Curso` são `1`, `2` e `3`) e na tabela acção o ID da `Accao` 1.

NOTA: Para os passos que se seguem vamos assumir que o ID da acção é o 33.

3. Queremos obter os id dos estudantes, classificação e data de inscrição de todos os estudantes da acção 1 do curso de '`Contabilidade`'. Introduza o seguinte comando:

```
SELECT IDEstudante, Classificacao, DataInscricao
FROM   Inscricao
WHERE  IDAccao = 33;
```

*O comando **`SELECT`** permite consultar tabelas e vistas. Para especificarmos critérios de consulta, ie, para seleccionarmos as linhas pretendidas, devemos acrescentar a cláusula **`WHERE`** à nossa pesquisa. Após esta palavra, colocamos os critérios de selecção. Para tal utilizamos operadores que "filtram" as linhas. Esses operadores podem ser o =, <> (diferente), >, >=, <, <=, **`BETWEEN`**, **`LIKE`**, etc. Podemos incluir vários critérios, ie, vários filtros. Esses filtros devem ser unidos através dos operadores lógicos **`AND`** ou **`OR`**. Por exemplo, suponhamos que pretendemos seleccionar todos os estudantes cujo código postal seja superior a 3002, excepto o 3005:*

```
SELECT * FROM Estudante WHERE CodigoPostal >= 3002 AND CodigoPostal <> 3005;
```

Ligações: http://pt.wikipedia.org/wiki/SQL .

4. Queremos obter os nomes, endereços e datas de nascimento de todos os alunos residentes em Lisboa ou Coimbra. Introduza o seguinte comando:

```
SELECT Nome, Endereco, DataNascimento
FROM   Estudante
WHERE  Cidade =  'Lisboa' OR Cidade = 'Coimbra';
```

5. A cláusula `WHERE` anterior também podia (e devia, uma vez que é mais simples e legível) ser escrita assim:

```
WHERE Cidade IN ('Lisboa', 'Coimbra');
```

6. Se todos os estudantes fossem obrigatoriamente de Lisboa, Coimbra ou Porto, então poderíamos escrever a seguinte cláusula `WHERE`:

```
WHERE  Cidade <> 'Porto';
```

7. Agora pretendemos obter todos os `Estudante`s cujo endereço começa pela palavra `'Rua'`. Introduza o seguinte comando:

```
SELECT Nome, Endereco
FROM   Estudante
WHERE  Endereco LIKE 'Rua%';
```

*O operador **LIKE** permite especificar um padrão de pesquisa em colunas de texto. Neste caso indicamos, através do caractere %, embebido no texto, que queremos todos os endereços que comecem por 'Rua'. Esta pesquisa devolve os endereços 'Rua B', 'Rua ', 'Rua', etc.. Não devolve os endereços que não comecem por Rua: ' Rua', 'A Rua', etc..*
Outro caractere de controlo é o _ , que indica um caractere qualquer nessa posição. Vamos supor que pretendemos todos os estudantes cujo código postal tenha 4 dígitos e comece por '300':

```
SELECT * FROM Estudante WHERE CodigoPostal LIKE '300_';
```

8. Introduza o seguinte comando para criar uma consulta que devolva todos os estudantes cujos códigos postais estejam entre 3002 e 3005 (incluindo ambos os limites):

```
SELECT *
FROM   Estudante
WHERE  CodigoPostal >= 3002 AND CodigoPostal <= 3005;
```

Neste tipo de consultas podemos, também, utilizar o operador `BETWEEN`:

```
WHERE  CodigoPostal BETWEEN 3002 AND 3005;
```

9. Para obter todos os estudantes cujos códigos postais estejam fora do intervalo anterior pode introduzir um dos seguintes comandos:

```
SELECT *
FROM   Estudante
WHERE  CodigoPostal < 3002 OR CodigoPostal > 3005;

SELECT *
FROM   Estudante
WHERE  NOT (CodigoPostal >= 3002 AND CodigoPostal <= 3005);

SELECT *
FROM   Estudante
WHERE  CodigoPostal NOT BETWEEN 3002 AND 3005;
```

10. Vamos agora seleccionar todos os alunos nascidos no ano de 1998. Como? Através do seguinte comando:

```
SELECT Nome, Apelido, DataNascimento
FROM   Estudante
WHERE  DataNascimento BETWEEN '1998-01-01' AND '1998-12-31';
```

11. Em alternativa poderia inserir algo como:

```
SELECT Nome, Apelido, DataNascimento
FROM   Estudante
WHERE  YEAR(DataNascimento) = 1998;
```

*O MySQL suporta funções. Uma **função** é um bloco de código parametrizado que devolve um ou mais valores. No contexto de 'queries' SQL, as funções são habitualmente utilizadas para extrair ou manipular os valores das colunas. Neste caso, recorremos à função **YEAR** para extrair o ano de uma coluna do tipo **DATE**.*

12. Continuando no domínio das datas, suponhamos que pretendíamos obter o número de dias que decorreram desde a data de inscrição dos estudantes respectivos em `Agricultura Aplicada 2` (ID assumido: 1):

```
SELECT IDEstudante, DATEDIFF(CURDATE(), DataInscricao)
FROM   Inscricao
WHERE  IDAccao = 1;
```

*A função **DATEDIFF (DT1, DT2)** devolve o número de dias entre as datas DT1 e DT2.*
*Por seu turno, com a palavra-reservada **INTERVAL** podemos indicar um intervalo de tempo. A síntaxe de utilização é:*

***INTERVAL** expr unidade*

*onde expr é uma expressão e unidade uma unidade de tempo como **YEAR**, **MONTH**, **DAY**, **WEEK**, etc..*
*As funções **DATE_ADD** e **DATE_SUB** também permitem adicionar/subtrair um intervalo de tempo a um valor do tipo **DATE** ou **DATETIME**.*

13. E agora insira a seguinte pesquisa para obter todas as inscrição nos últimos 14 meses:

```
SELECT IDEstudante, DataInscricao
FROM   Inscricao
WHERE  DataInscricao >= CURDATE() - INTERVAL 14 MONTH;
```

14. Também podemos saber o dia do ano, do mês e da semana:

```
SELECT IDEstudante,
       DAYOFYEAR(DataInscricao),
       DAYOFMONTH(DataInscricao),
       DAYOFWEEK(DataInscricao)
FROM Inscricao;
```

2a Parte – Manipulação das colunas resultantes das consultas

15. Vamos refazer uma das consulta anteriores. A partir das colunas `Nome` e `Apelido`, vamos devolver uma coluna com o nome completo dos estudantes. A função **`CONCAT`** concatena dois ou mais valores. Vamos usá-la para concatenar as duas colunas mencionadas. A palavra reservada **`AS`** serve para atribuir `"Nome completo"` a essa coluna:

```
SELECT CONCAT(Nome, Apelido) AS "Nome completo",
       DataNascimento  AS "Data de nascimento"
FROM   Estudante
WHERE  DataNascimento BETWEEN '1998-01-01' AND '1998-12-31';
```

16. Altere a consulta anterior de modo a que `Nome` e `Apelido` não apareçam "colados" na coluna `"Nome completo"`.

3a Parte – Vistas

17. Suponhamos que a consulta a respeito dos estudantes do curso de `'Contabilidade'` é muito frequente. Vamos criar uma vista para "guardar" essa consulta. Introduza o seguinte comando:

```
CREATE VIEW EstContab AS
  SELECT IDEstudante,
         DataInscricao
  FROM   Inscricao
  WHERE  IDAccao IN (33, 22);
```

*Uma vista é uma tabela que é "calculada" no momento que se pretende aceder a essa "tabela". Ou seja, uma vista é uma consulta com um nome e cria-se através do comando SQL **`CREATE VIEW`** `<nome da vista>` **`AS`** `<comando select>`. Uma vez criada a vista, ela pode ser utilizada tal como se de uma tabela normal se tratasse.*

18. Agora, seleccione todos os alunos de `'Contabilidade'` através da vista, introduzindo o seguinte comando:

```
SELECT * FROM EstContab;
```

19. Crie uma vista sobre os estudantes dos cursos de natureza e dê-lhe o nome `EstNatureza`.

20. Crie uma vista, dando-lhe o nome `Maiores25`, sobre todos os estudantes com mais de 25 anos.

21. Crie uma consulta que devolva todos os estudantes maiores do que 25 anos cujo `Nome completo` comece por `'António'`.

EXERCÍCIOS

1. Utilizando o comando SELECT, desenvolva consultas para as seguintes situações:

1.1 Todos os estudantes de Contabilidade 2. Queremos obter o id, primeiro nome e idade aproximada (ie, diferença de anos entre a data actual e a data de nascimento).

1.2 Todos os estudantes nascidos entre 1998 e 2001. Queremos obter nome completo, data de nascimento e NISS. O nome completo deve estar no formato americano ("apelido, nome pessoal").

1.3 Todas as acções iniciadas em 2021. Pretende-se o número da acção e o coordenador.

1.4 Todos os estudantes nascidos em Fevereiro e cujo nome pessoal tenha 6 ou mais letras. Queremos obter todas as colunas.

1.5 Todos os cursos cuja duração termine em 0 (zero). Devolva o nome do curso e a duração.

2. Estude as funções para maninpulação de *strings*, e resolva os seguintes exercícios:

2.1 Qual a diferença entre as funções `LENGTH` e `CHAR_LENGTH`?

2.2 Indique uma consulta para obter os três primeiros caracteres do nome de cada estudante cujo código postal seja um número par.

2.3 Indique uma consulta para obter primeiro nome e apelido. O apelido deve ser devolvido numa coluna com 15 espaços alinhado à direita. Se o nome tiver menos do que 15 espaços, os espaços à esquerda devem ser preenchidos com `'_'` (*underscore*).

2.4 A coluna `Apelido` da tabela `Estudante` pode armazenar vários apelidos de família (separados por um espaço). Indique uma consulta para devolver apenas o primeiro nome e o último apelido de cada estudante.

3. Indique uma consulta para obter os ID dos estudantes inscritos há menos do que um ano.

4. Indique uma consulta para obter as idades de todos os estudantes. A função IF poderá ajudar...

5. Sabendo que a duração de cada curso será distribuída por períodos diários de 10h, indique uma consulta para devolver o nome do respectivo curso e a duração do curso no seguinte formato: 'XdYh'. Por exemplo, se o curso de `'Mecânica p/ Curiosos'` tiver uma duração de `25h`, a sua consulta deverá devolver algo parecido com: `'Mecânica p/ Curiosos' | '2d5h'`.

6. Crie uma vista para indicar o nome de cada estudante, a cidade, a morada, o código postal e a região de cada um dos estudantes. Para efeitos deste exercício (e por simplicidade), um `Estudante` é do `'NORTE'` se a sua cidade for uma das seguintes: Braga, Guimarães, Porto, Aveiro e Viseu; é do `'CENTRO'` se for uma das seguintes: Coimbra, Leiria, Tomar, Lisboa; é do `SUL` se for uma das seguintes: Setúbal, Évora, Beja, Faro; é de `'OUTRA_REGIÃO'`, se não for de nenhuma das outras.

A região deve ser indicada numa coluna extra a devolver pela consulta que implementa a vista. Designe a vista de `RegiaoEstudantes`. Desenvolva uma primeira versão utilizando a expressão/função `IF` e uma segunda versão com a expressão/função `CASE-WHEN` em vez do `IF`.