21053 - Fundamentos de Bases de Dados
2011-2012
e-fólio A
Resolução e Critérios de Correcção

PARA A RESOLUÇÃO DO E-FÓLIO, ACONSELHA-SE QUE LEIA ATENTAMENTE O SEGUINTE:

1) O e-fólio é constituído por 2 perguntas, num total de 10 alíneas, com cotação 0,2 valor cada. A cotação global é de 2 valores.

2) O e-fólio deve ser entregue num único ficheiro PDF, não zipado, com fundo branco, com perguntas numeradas e sem necessidade de rodar o texto para o ler. Penalização de 1 a 2 valores.

3) Não são aceites e-fólios manuscritos, i.e. tem penalização de 100%.

4) O nome do ficheiro deve seguir a normal "eFolioA" + <nº estudante> + <nome estudante com o máximo de 3 palavras>

5) Durante a realização do e-fólio, os estudantes devem concentrar-se na resolução do seu trabalho individual, não sendo permitida a colocação de perguntas ao professor ou entre colegas.

6) A interpretação das perguntas também faz parte da sua resolução, se encontrar alguma ambiguidade deve indicar claramente como foi resolvida.

7) A legibilidade, a objectividade e a clareza nas respostas serão valorizadas, pelo que, a falta destas qualidades serão penalizadas.

1) Considere a Base de Dados de Encomendas de Produtos: crie as tabelas, ligações entre tabelas e respectivo carregamento de dados para testes. De seguida crie consultas em SQL para as alíneas que se seguem:

cliente (cliente_id -> nome, morada, código postal, telefone, e-mail)
produto (produto_id -> nome, preço, unidade)
encomenda (encomenda_id -> data, cliente_id, prazo, satisfeito <s/n>)
linha_encomenda (encomenda_id, linha -> produto_id, quantidade)

1.1) Contar as encomendas que já foram satisfeitos.

1.2) Seleccionar todos os produtos cujo nome possua a string 'an' em qualquer posição do nome.

1.3) Mostrar a identificação da encomenda, o nome do produto e valor da linha (preço x quantidade), das encomendas cuja soma total seja superior a 100 euros.

1.4) Utilizando a cláusula IN responda: quais os produtos que nunca foram encomendados?

1.5) Utilizando a cláusula EXISTS responda: quais os clientes que nunca fizeram encomendas?

1.6) Mostrar os dados dos produtos que nunca foram comprados por clientes que moram em Lisboa.

1.7) Mostrar os dados dos clientes que compraram no dia 02/12/2010 e que não compraram no dia 04/12/2010.

1.8) Mostrar identificador, nome e preço dos produtos cujo preço do produto seja maior que a média de preço de todos os produtos.

2) Utilize a mesma base de dados da alínea anterior, consulte http://www.w3schools.com/SQl/default.asp e responda às seguintes perguntas:
2.a) Exemplifique uma Junção à Esquerda;
2.b) Exemplifique uma Junção à Direita.

Comentário e Critérios de Correcção Gerais:
- consultas SQL não precisam de ser comentadas
- falta de indentação das consultas SQL: -0,1 valores
- consultas SQL numa única linha: -0,1 ou -0,2 se ilegível

1.1) Contar as encomendas que já foram satisfeitos.

```
SELECT Count(encomenda.encomenda_id)
FROM encomenda
WHERE encomenda.satisfeito<>False
```

1.2) Seleccionar todos os produtos cujo nome possua a string 'an' em qualquer posição do nome.

```
SELECT produto.produto_id
FROM produto
WHERE produto.produto_id LIKE "*an*"
```

1.3) Mostrar a identificação da encomenda, o nome do produto e valor da linha (preço x quantidade), das encomendas cuja soma total seja superior a 100 euros.

```
SELECT encom_id,prod_id Sum([encomendalinha].[quantidade]*[produto].[preço])
FROM produto, encomenda-linha
WHERE (((produto.produto_id)=[encomenda-linha].[produto_id]))
GROUP BY [encomenda-linha].encomenda_id
HAVING (((Sum([encomenda-linha].[quantidade]*[produto].[preço]))>100));
```

Critério de correcção: (-0,2 valores) se não usar Having
Qual a diferença entre as cláusulas Where e Having?
- a cláusula Where manipula linhas (tuplos)
- a cláusula Having manipula informação dos grupos

1.4) Utilizando a cláusula IN responda: quais os produtos que nunca foram encomendados?

```
SELECT produto.produto_id
FROM produto
WHERE (((produto.produto_id)  NOT IN (SELECT [encomenda-linha].produto_id
                                      FROM [encomenda-linha])));
```

1.5) Utilizando a cláusula EXISTS responda: quais os clientes que nunca fizeram encomendas?

```
SELECT cliente.cliente_id
FROM cliente
WHERE NOT EXISTS (SELECT *
                  FROM encomenda
                  WHERE encomenda.cliente_id=cliente.cliente_id)
```

1.6) Mostrar os dados dos produtos que nunca foram comprados por clientes que moram em Lisboa.

1.6 auxiliar) produtos comprados por clientes de Lisboa

```
SELECT [encomenda-linha].produto_id, cliente.cliente_id, cliente.[código postal]
FROM cliente, encomenda, [encomenda-linha]
WHERE (((cliente.cliente_id)=[encomenda].[cliente_id])
AND ((encomenda.encomenda_id)=[encomenda-linha].[encomenda_id]))
AND ((cliente.[código postal]) Like "*lisboa")
```

Resposta da 1.6)

```
SELECT produto.produto_id, produto.preço, produto.unidade
FROM produto
WHERE produto.produto_id NOT IN
        ( SELECT [encomenda-linha].produto_id
          FROM cliente, encomenda, [encomenda-linha]
          WHERE (((cliente.cliente_id)=[encomenda].[cliente_id])
          AND ((encomenda.encomenda_id)=[encomenda-linha].[encomenda_id]))
          AND ((cliente.[código postal]) Like "*lisboa") )
```

Critério de correcção: (-0,1 valores) se usar cadeias de sub-consultas; a junção é geralmente mais eficiente que as sub-consultas.

1.7) Mostrar os dados dos clientes que compraram no dia 02/12/2010 e que não compraram no dia 04/12/2010.

```
SELECT cliente_id
FROM cliente
WHERE cliente_id IN (SELECT DISTINCT cliente_id
                     FROM encomenda
                     WHERE data = #12/02/2010#)
AND cliente_id NOT IN (SELECT DISTINCT cliente_id
                       FROM encomenda
                       WHERE data = #12/04/2010#);
```

1.8) Mostrar identificador, nome e preço dos produtos cujo preço do produto seja maior que a média de preço de todos os produtos.

```
SELECT produto_id, nome, [preço]
FROM produto
WHERE produto.[preço] > (SELECT AVG(produto.[preço]) FROM produto);
```

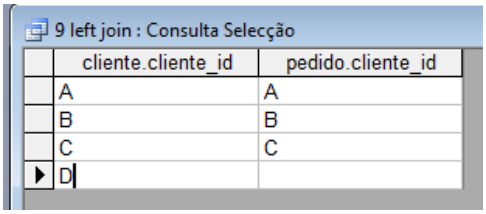
Critério de correcção: (-0,1 valores) se usar ALL

Qual a diferença entre:
WHERE produto.preco > (SELECT AVG(produto.preco) FROM produto) e
WHERE produto.preco > ALL (SELECT AVG(produto.preco) FROM produto)
- a cláusula ALL manipula um conjunto de linhas (tuplos); no nosso caso a função AVG devolve uma única linha, pelo que ALL é desnecessário.

2) Utilize a mesma base de dados da alínea anterior, consulte http://www.w3schools.com/SQl/default.asp e responda às seguintes perguntas:
2.a) Exemplifique uma Junção à Esquerda;
2.b) Exemplifique uma Junção à Direita.

2.a) Exemplifique uma Junção à Esquerda

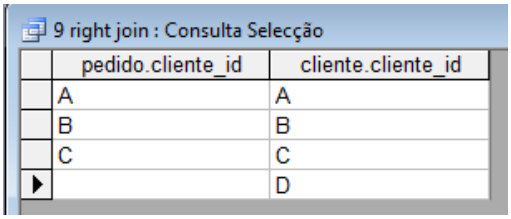
```
SELECT DISTINCT cliente.cliente_id, pedido.cliente_id
FROM cliente LEFT JOIN pedido ON cliente.cliente_id = pedido.cliente_id
```


9 left join : Consulta Selecção

cliente.cliente_id	pedido.cliente_id
A	A
B	B
C	C
D	

2.b) Exemplifique uma Junção à Direita

```
SELECT DISTINCT pedido.cliente_id, cliente.cliente_id
FROM pedido RIGHT JOIN cliente ON pedido.cliente_id = cliente.cliente_id
```


9 right join : Consulta Selecção

pedido.cliente_id	cliente.cliente_id
A	A
B	B
C	C
	D