Universidade do Minho
Departamento de Informática

Curso: Licenciatura em Engenharia Informática (LEI);
Mestrado Integrado em Engenharia Informática (MIEI)
Unidade Curricular: Bases de Dados
Ano Letivo: 2021/2022 – 2º Semestre

FICHA DE EXERCÍCIOS – PL13

Questão 1. Crie a função idade para calcular a diferença em anos entre a data atual e um determinado ano, caso ainda não tenha criado a função idade.

```
DELIMITER //

CREATE FUNCTION idade(dta DATE) RETURNS INT

DETERMINISTIC

BEGIN

  RETURN TIMESTAMPPDIFF(YEAR, dta, CURDATE());

END //

DELIMITER ;
```

Questão 2. Considere que a administração do hospital pediu à EXIT um novo requisito, nomeadamente, que o preço das consultas de cada especialidade fosse tabelado. Sendo assim, altere a tabela *especialidades* do caso prático de modo a incluir o atributo "*preço*".

```
ALTER TABLE especialidades ADD preco DECIMAL(5,2) NOT NULL DEFAULT 10;
```

Questão 3. Crie um procedimento que permita atualizar o preço de uma determinada especialidade.

```
DELIMITER $$

CREATE PROCEDURE NewPrecoEspecialidade (IN cod_esp INT, IN new_preco
DECIMAL(5,2))

BEGIN

  UPDATE especialidades SET preco = new_preco where cod_especialidade=cod_esp;

END $$

DELIMITER ;
```

Questão 4. Atualize os valores da coluna *preco* da tabela *consultas* com base nos valores de cada especialidade mais os eventuais procedimentos realizados.

```
-- caso o SQL_SAFE_UPDATES esteja ativo significa que não é possível atualizar ou excluir registos sem especificar uma chave (por exemplo, chave primária) na cláusula where. Por isso, vamos desativar esta opção temporariamente.

SET SQL_SAFE_UPDATES = 0;

-- update do preço total das consultas

UPDATE consultas co,

(SELECT e.preco AS c_preco, p.preco AS p_preco, c.nr_episodio FROM especialidades e

NATURAL JOIN medicos m

INNER JOIN consultas c ON m.num_mec=c.id_medico

LEFT JOIN procedimentos p ON p.cod_procedimento=c.id_proc

) s

SET co.preco = CASE

      WHEN s.p_preco IS NULL THEN s.c_preco

      ELSE s.c_preco + s.p_preco

END

WHERE co.nr_episodio=s.nr_episodio;

-- caso o SQL_SAFE_UPDATES tenha sido desativado, vamos voltar a ativar esta opção por medida de segurança.

SET SQL_SAFE_UPDATES = 1;
```

Questão 5. Considere que a administração do hospital pediu à EXIT mais um requisito. Desta vez, pediram que fossem incluídos o número total de consultas por especialidade (*nr_consultas*) e o valor total faturado das consultas por especialidade (*total_faturado*), excluindo o valor de eventuais procedimentos que possam ter ocorrido. Desenvolva as instruções SQL que permitem acomodar estas alterações. Com base nos dados já registados na BD, atualize os valores destas colunas para cada especialidade.

```
-- primeiro é necessário adicionar as novas colunas

ALTER TABLE especialidades

  ADD nr_consultas INT NOT NULL DEFAULT 0,

  ADD total_faturado DECIMAL(5,2) NOT NULL DEFAULT 0;
```

```
-- caso o SQL_SAFE_UPDATES esteja ativo significa que não é possível atualizar ou excluir registos sem especificar uma chave (por exemplo, chave primária) na cláusula where. Por isso, vamos desativar esta opção temporariamente.
SET SQL_SAFE_UPDATES = 0;
-- update do número total de consultas
UPDATE especialidades e,
(SELECT e.des_especialidade, count(*) AS total FROM consultas c
INNER JOIN medicos m ON c.id_medico=m.num_mec
INNER JOIN especialidades e ON e.cod_especialidade=m.cod_especialidade
GROUP BY e.des_especialidade) n
SET e.nr_consultas=n.total
WHERE e.des_especialidade=n.des_especialidade;
```

Esta *subquery* também poderia ser:

```
SELECT e.des_especialidade, count(*) AS total FROM consultas c
INNER JOIN medicos m ON c.id_medico=m.num_mec
INNER JOIN especialidades e USING(cod_especialidade)
GROUP BY e.des_especialidade;
```

ou

```
SELECT e.des_especialidade, count(*) as nr_consultas FROM especialidades e
NATURAL JOIN medicos m
INNER JOIN consultas c ON c.id_medico=m.num_mec
GROUP BY e.des_especialidade;
```

```
-- update do valor total faturado
UPDATE especialidades e,
(SELECT e.des_especialidade, SUM(e.preco) as total FROM especialidades e
NATURAL JOIN medicos m INNER JOIN consultas c ON c.id_medico=m.num_mec
GROUP BY e.des_especialidade) n
SET e.total_faturado=n.total
WHERE e.des_especialidade=n.des_especialidade;
-- caso o SQL_SAFE_UPDATES tenha sido desativado, vamos voltar a ativar esta opção por medida de segurança.
SET SQL_SAFE_UPDATES = 1;
```

Questão 6. Crie um *trigger* para manter as colunas criadas na alínea anterior sempre atualizadas. Responda à questão tendo em consideração que a administração do hospital definiu que não é possível remover consultas nem especialidades da Base de Dados.

```
-- como não é possível remover especialidades nem consultas, resta verificar se a adição ou atualização de alguma destas tabelas terá influência nos valores das colunas

-- a atualização de consultas não influencia o nr_consultas nem o valor_faturado

-- a inserção de consultas influencia o nr_consultas e o valor_faturado

DELIMITER $$

CREATE TRIGGER AtualizarEspecialidadesInsertConsulta

AFTER INSERT

ON consultas FOR EACH ROW

BEGIN

  UPDATE especialidades e, medicos m

  SET e.nr_consultas = e.nr_consultas + 1,

  e.total_faturado = e. total_faturado + e.preco

  WHERE m.cod_especialidade=e.cod_especialidade

  AND m.num_mec=new.id_medico;

END $$

DELIMITER ;

-- a inserção de novas especialidades não influencia o nr_consultas nem o valor_faturado, já que definimos que por default estas colunas possuem o valor 0. Só quando for inserida uma consulta da nova especialidade, é que os valores vão ser alterados. Neste caso, a atualização já é garantida pelo trigger AtualizarEspecialidadesInsertConsulta.

-- a atualização de especialidades não influencia o nr_consultas nem o valor_faturado atual mesmo que o preço seja alterado, pois a alteração de preço não deve influenciar o valor de faturação das consultas que ocorreram antes da alteração da tabela de preços. Depois quando for inserida uma nova consulta, após a alteração da tabela de preços, será aplicado o valor de preço atual pelo trigger AtualizarEspecialidadesInsertConsulta.
```