PL7 - SQL DML SELECT – SUBQUERIES (NÃO-CORRELACIONADAS E CORRELACIONADAS)

Category	Exercícios
Files	PL07-Files.zip PL07-SQL DML SELECT, subqueries-PT.pdf

▼ Quick Access

▼ Create

```
-- ** eliminar tabelas se existentes **
-- CASCADE CONSTRAINTS para eliminar as restrições de inte
-- PURGE elimina a tabela da base de dados e da "reciclage
DROP TABLE marinheiro  CASCADE CONSTRAINTS PURGE;
DROP TABLE barco       CASCADE CONSTRAINTS PURGE;
DROP TABLE reserva     CASCADE CONSTRAINTS PURGE;

-- ## tabela Marinheiro ##
CREATE TABLE marinheiro(
  idMarinheiro      INTEGER     CONSTRAINT pkMarinheiroIdM
  nome              VARCHAR(30) CONSTRAINT nnMarinheiroNom
  classificacao     INTEGER     CONSTRAINT nnMarinheiroCla
  dataNascimento    DATE        CONSTRAINT nnMarinheiroDat
);

-- ## tabela Barco ##
CREATE TABLE barco(
  idBarco       INTEGER     CONSTRAINT pkBarcoIdBarco
  designacao    VARCHAR(20) CONSTRAINT nnBarcoDesignacao
  cor           VARCHAR(10) CONSTRAINT nnBarcoCor
```

```
);

-- ## tabela Reserva ##
CREATE TABLE reserva(
  idMarinheiro  INTEGER,
  idBarco       INTEGER,
  data          DATE        CONSTRAINT nnReservaData  NOT 

  CONSTRAINT pkReservaIdMarinheiroIdBarcoData PRIMARY KEY(
);

-- ** alterar tabelas para definição de chaves estrangeira
ALTER TABLE reserva ADD CONSTRAINT fkReservaIdMarinheiro F
ALTER TABLE reserva ADD CONSTRAINT fkReservaIdBarco      F

-- ** guardar em DEFINITIVO as alterações na base de dados
-- COMMIT;
```

▼ Inserts

```
-- ** inserir dados nas tabelas **

-- ## tabela Marinheiro ##
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 
INSERT INTO marinheiro(idMarinheiro, nome, classificacao, 

-- ## tabela Barco ##
INSERT INTO barco(idBarco, designacao, cor) VALUES(101, 'I
```

```
INSERT INTO barco(idBarco, designacao, cor) VALUES(102, 'I
INSERT INTO barco(idBarco, designacao, cor) VALUES(103, 'C
INSERT INTO barco(idBarco, designacao, cor) VALUES(104, 'M

-- ## tabela Reserva ##
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(22
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(22
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(22
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(22
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(31
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(31
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(31
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(64
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(64
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(64
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(64
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(74
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(44
INSERT INTO reserva(idMarinheiro, idBarco, data) VALUES(44

-- ** guardar em DEFINITIVO as alterações na base de dados
-- COMMIT;
```

A. Operações sem recorrer ao uso de subqueries correlacionadas:

▼ 1) Mostrar o id, o nome e a classificação dos marinheiros com a melhor classificação, usando três estratégias diferentes, nomeadamente:

▼ a) Sem junção de tabelas e com função de agregação;

```
SELECT idMarinheiro, nome, classificacao
FROM marinheiro
ORDER BY classificacao DESC
FETCH FIRST 3 ROW ONLY;
```

▼

IDMARINHEIRO	NOME	CLASSIFICACAO
58	Rusty	10
71	Zorba	10
74	Horácio	9
31	Lubber	8
32	Andy	8

▼ b) Sem junção de tabelas e sem função de agregação;

```
SELECT idMarinheiro, nome, classificacao
FROM marinheiro
WHERE classificacao = (SELECT MAX(classificacao) FROM m
```

▼

IDMARINHEIRO	NOME	CLASSIFICACAO
58	Rusty	10
71	Zorba	10

▼ c) Com junção de tabelas.

```
SELECT m1.idMarinheiro, m1.nome, m1.classificacao
FROM marinheiro m1
INNER JOIN marinheiro m2 ON m1.classificacao = m2.class
WHERE m1.idMarinheiro < m2.idMarinheiro;
```

▼

IDMARINHEIRO	NOME	CLASSIFICACAO
31	Lubber	8
22	Dustin	7
58	Rusty	10
13	Popeye	3
44	Haddock	3
85	Art	3
13	Popeye	3
44	Haddock	3
13	Popeye	3

▼ 2) Mostrar o id e o nome dos marinheiros que não reservaram barcos, usando três estratégias diferentes, nomeadamente:

▼ a) Com operador IN

```
SELECT idMarinheiro, nome
FROM marinheiro
WHERE idMarinheiro NOT IN (SELECT DISTINCT idMarinheiro
```

▼

IDMARINHEIRO	NOME
58	Rusty
85	Art
71	Zorba
29	Brutus
95	Bob
13	Popeye
32	Andy

▼ b) Com condição ANY

```
SELECT idMarinheiro, nome
FROM marinheiro
WHERE idMarinheiro <> ANY (SELECT idMarinheiro FROM res
```

▼

IDMARINHEIRO	NOME
22	Dustin
29	Brutus
31	Lubber
32	Andy
58	Rusty
64	Horácio
71	Zorba
74	Horácio
85	Art
95	Bob
13	Popeye
44	Haddock

▼ c) Com operador de conjuntos

```
SELECT idMarinheiro, nome
FROM marinheiro
WHERE NOT EXISTS (
    SELECT 1
    FROM reserva
    WHERE reserva.idMarinheiro = marinheiro.idMarinheir
);
```

▼ 3) Mostrar o número total de marinheiros que reservaram barcos com a cor vermelho e barcos com a cor verde

```
SELECT COUNT(DISTINCT r.idMarinheiro) AS totalMarinheiros
FROM reserva r
JOIN barco b ON r.idBarco = b.idBarco
WHERE b.cor IN ('vermelho', 'verde');
```

B. Operações para recorrer ao uso de subqueries correlacionadas:

▼ 4) Mostrar o id, o nome e a quantidade de reservas de barcos dos marinheiros registados na BD, por ordem decrescente da quantidade de reservas

```
SELECT m.idMarinheiro, m.nome, COUNT(r.idMarinheiro) AS qu
FROM marinheiro m
LEFT JOIN reserva r ON r.idMarinheiro = m.idMarinheiro
GROUP BY m.idMarinheiro, m.nome
ORDER BY quantidadeReservas DESC;
```

▼ 5) Mostrar o id dos marinheiros cuja quantidade de reservas de um barco seja superior à quantidade média de reservas desse barco. Além disso, o resultado deve também incluir o id do barco e a quantidade de reservas.

```
WITH ReservasPorBarco AS (
    SELECT idBarco, COUNT(*) AS quantidadeReservas
    FROM reserva
    GROUP BY idBarco
),
MediaReservasporBarco AS (
    SELECT AVG(quantidadeReservas) AS mediaReservas
    FROM ReservasPorBarco
)
SELECT r.idMarinheiro, r.idBarco, rb.quantidadeReservas
FROM reserva r
JOIN ReservasPorBarco rb ON r.idBarco = rb.idBarco
JOIN MediaReservasPorBarco mr ON 1 = 1
WHERE rb.quantidadeReservas > mr.mediaReservas;
```

▼

IDMARINHEIRO	IDBARCO	QUANTIDADERESERVAS
22	101	4
22	102	5
31	102	5
44	101	4
44	101	4
64	101	4
64	102	5
64	102	5
64	102	5