INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e de Gestão

Bases de Dados
2025/26

Caderno de Exercícios

SQL

@jprp | 2025

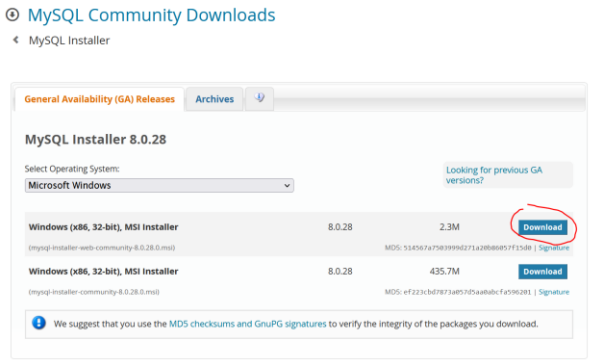
Índice

Exercício 1 | Instalar MySQL

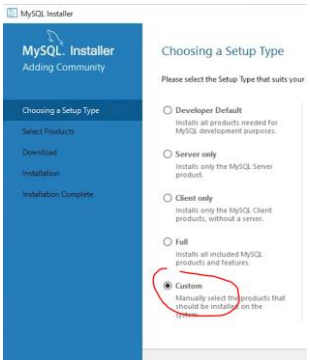
1) Instalar MySQL

a) download

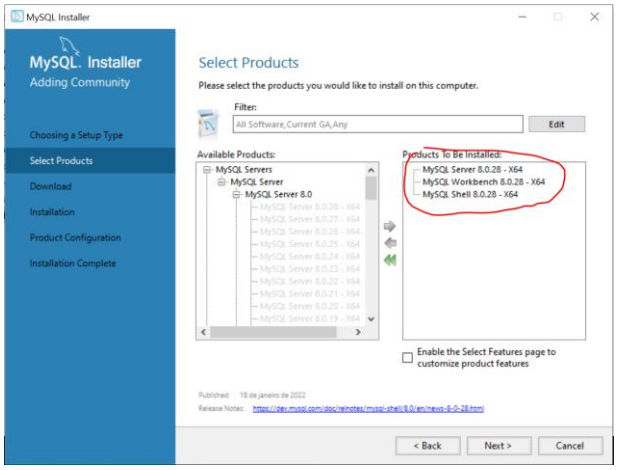
https://dev.mysql.com/downloads/installer/



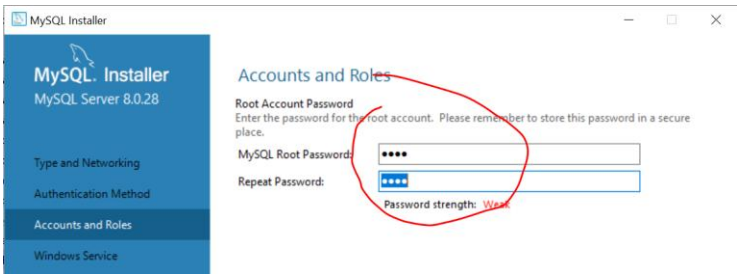
b) Escolher a opção "custom"



c) Selecionar: server, Workbench e shell

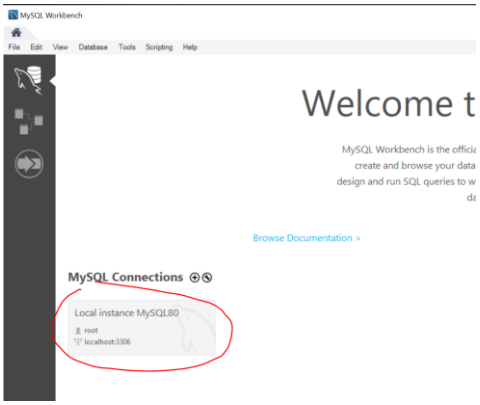


d) Definir password



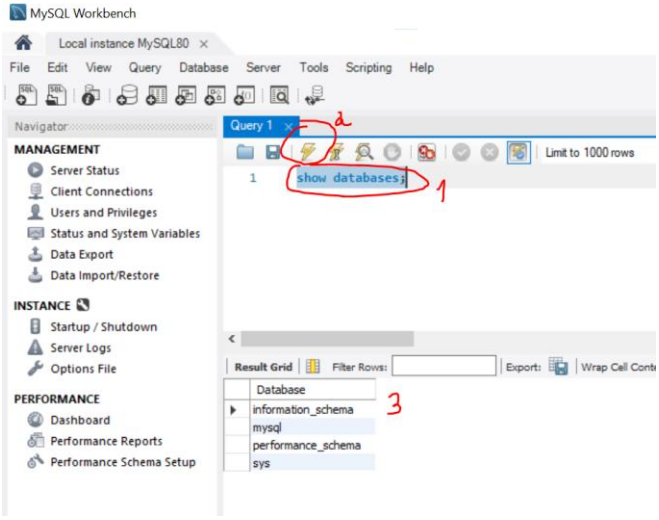
2) Configurar servidor

a) Clicar em "Local instance"



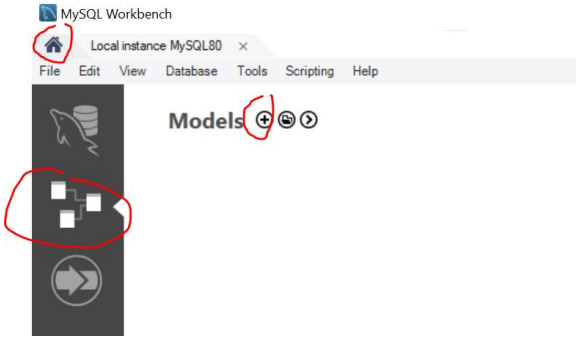
b) Inserir password escolhida na instalação

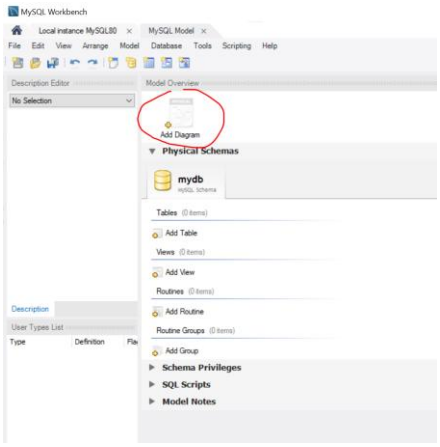
3) Testar



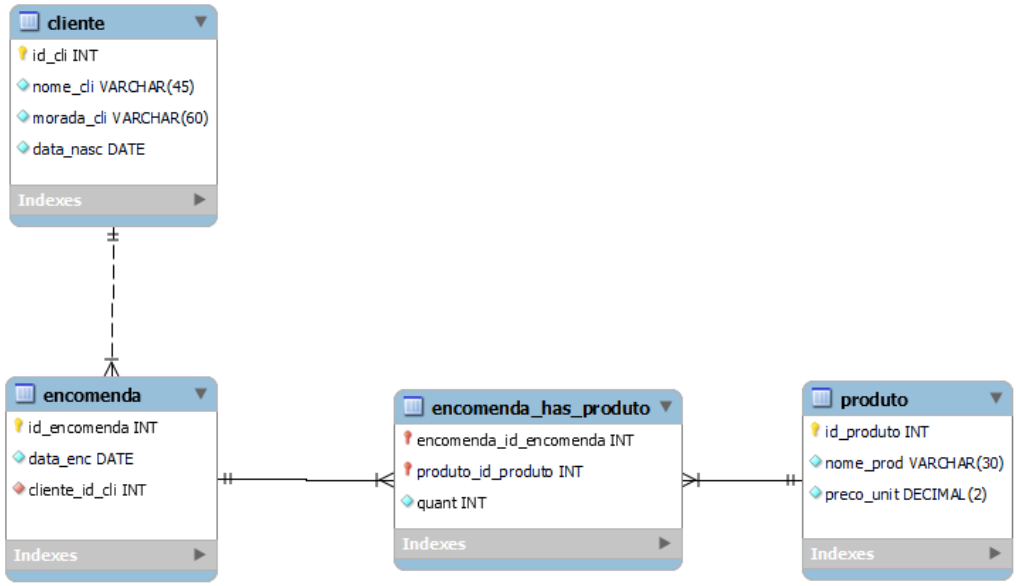
Exercício 2 | Criar modelo relacional

1) Criar um novo modelo

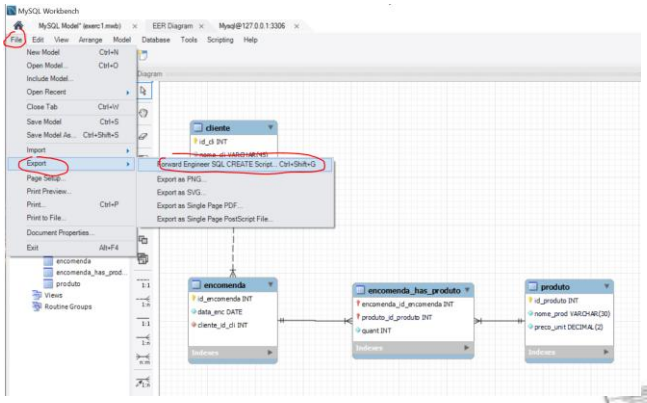




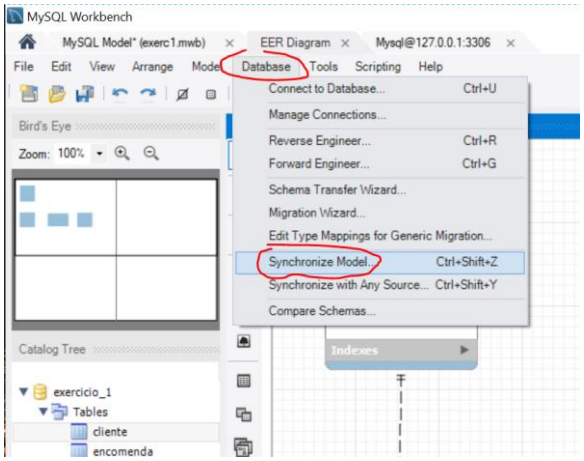
2) Criar o seguinte modelo relacional



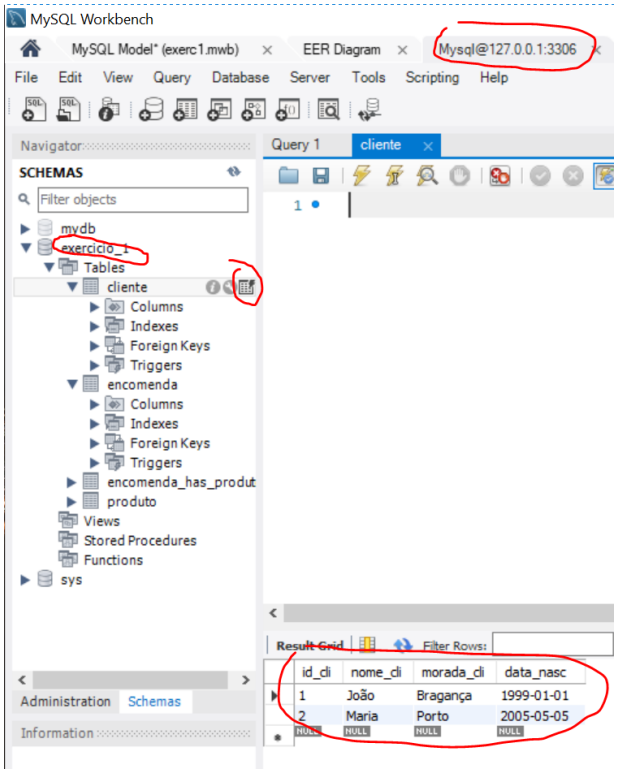
3) Exportar SQL



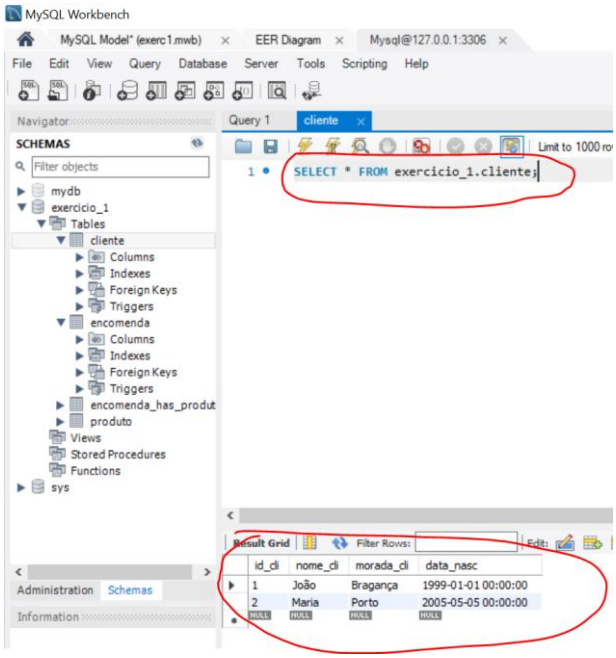
4) Mudar o nome do esquema para “exercício_1” e sincronizar com o servidor



5) Inserir registos no servidor



6) Verificar se os dados foram inseridos na tabela “cliente”


MySQL Workbench
MySQL Model* (exerc1.mwb)
EER Diagram
Mysql@127.0.0.1:3306
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
mydb
exercicio_1
Tables
cliente
Columns
Indexes
Foreign Keys
Triggers
encomenda
Columns
Indexes
Foreign Keys
Triggers
encomenda_has_produt
produto
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Query 1
cliente
SELECT * FROM exercicio_1.cliente;
Result Grid
Filter Rows:
id_cli nome_cli morada_cli data_nasc
1 João Bragança 1999-01-01 00:00:00
2 Maria Porto 2005-05-05 00:00:00

Exercício 3 | SQL / base de dados "Aulas"

1) Com base no SQL seguinte, desenhar o diagrama relacional.

2) No MySQL, criar a base de dados "AULAS" e respectivas tabelas

a) Criação da tabela Postal.

```
CREATE TABLE Postal
             (Codigo  Numeric(4)   PRIMARY KEY,
              Local   Char(30) NOT NULL );
```

b) Criação da Tabela Pessoa

```
CREATE TABLE Pessoa
    (Id NUMERIC           PRIMARY KEY,
    Nome CHAR(30)          NOT NULL,
    Idade  INTEGER           NOT NULL CHECK  (Idade BETWEEN 0 AND 150),
    Salario  NUMERIC(10,2)     NOT NULL CHECK (Salario>0),
    Telefone  CHAR(12)          NULL,
    Cod_Postal NUMERIC(4)
           REFERENCES Postal(Codigo)
       ON UPDATE CASCADE
       ON DELETE CASCADE);
```

c) Criação da Tabela Mensagem

```
CREATE TABLE Mensagem
        (Id_Msg   NUMERIC           PRIMARY KEY,
         Mensagem   CHAR(30)          NOT NULL );
```

d) Criação da Tabela Comissao

```
CREATE TABLE Comissao
              (Id       NUMERIC          NOT NULL,
               Id_Msg   NUMERIC          NOT NULL,
               Valor    NUMERIC          NOT NULL );
```

3) Inserção de valores

a) tabela postal.

```
INSERT INTO Postal VALUES(1000,'LISBOA');
INSERT INTO Postal VALUES(1100,'LISBOA');
INSERT INTO Postal VALUES(1200,'LISBOA');
INSERT INTO Postal VALUES(1500,'LISBOA');
INSERT INTO Postal VALUES(2000,'SANTAREM');
INSERT INTO Postal VALUES(2300,'TOMAR');
INSERT INTO Postal VALUES(3000,'COIMBRA');
INSERT INTO Postal VALUES(4000,'PORTO');
INSERT INTO Postal VALUES(9000,'FUNCHAL');
```

b) tabela Pessoa.

```
INSERT INTO Pessoa VALUES(42,'António Dias',43,74000,'789654',1500);
INSERT INTO Pessoa VALUES(5,'Célia Morais',26,170000,'123456',1100);
INSERT INTO Pessoa VALUES(32,'Florinda Simões',35,147000,NULL,4000);
```

```
INSERT INTO Pessoa VALUES(37,'Isabel Espada',28,86000,NULL,1100);
INSERT INTO Pessoa VALUES(49,'José António',17,210000,NULL,1500);
INSERT INTO Pessoa VALUES(14,'Nascimento Augusto',35,220000,'456123',2300);
INSERT INTO Pessoa VALUES(25,'Paulo Viegas',32,95000,NULL,1500);
```

c) tabela Mensagem.

```
INSERT INTO Mensagem VALUES(10,'Comissão de Vendas');
INSERT INTO Mensagem  VALUES(20,'Fretes Individuais');
INSERT INTO Mensagem  VALUES(30,'Fretes Empresas');
INSERT INTO Mensagem VALUES(40,'Vendas Extra');
INSERT INTO Mensagem VALUES(50,'Deslocações');
INSERT INTO Mensagem VALUES(60,'Refeições');
INSERT INTO Mensagem VALUES(70,'Combustíveis');
INSERT INTO Mensagem VALUES(80,'Transportes');
INSERT INTO Mensagem VALUES(90,'Telefonemas');
INSERT INTO Mensagem VALUES(100,'Ofertas');
```

d) tabela Comissão.

```
INSERT INTO Comissao VALUES(14,10,10500);
INSERT INTO Comissao VALUES(25,10,2500);
INSERT INTO Comissao VALUES(14,100,3750);
INSERT INTO Comissao VALUES(14,70,400);
INSERT INTO Comissao VALUES(37,40,20);
INSERT INTO Comissao VALUES(37,30,14230);
INSERT INTO Comissao VALUES(37,10,5500);
INSERT INTO Comissao VALUES(14,60,2600);
INSERT INTO Comissao VALUES(25,30,370);
INSERT INTO Comissao VALUES(40,20,20);
INSERT INTO Comissao VALUES(37,50,120);
INSERT INTO Comissao VALUES(42,20,170);
INSERT INTO Comissao VALUES(49,20,2300);
```

4) Seleccionar todos os campos da tabela Pessoa

```
SELECT *
FROM Pessoa;
```

5) Seleccionar apenas o campo Nome, Salario e Cod_postal da tabela pessoa

```
SELECT Nome, salario, Cod_postal
FROM Pessoa;
```

6) Seleccionar Nome, Idade e Salário das pessoas com 35 anos

OPREADORES RELACIONAIS
[=, >, >, >=, <=, <> ou !=]

```
SELECT Nome, Idade, salario
FROM Pessoa
WHERE Idade = 35;
```

7) Seleccionar todos os campos das pessoas com idade igual ou superior a 18 anos

```
SELECT *
FROM Pessoa
WHERE Idade >= 18;
```

OPERADORES lÓGICOS
[AND, OR, NOT]

8) Seleccionar Id, Nome e Idade das Pessoas com idade entre 30 e 40 anos

9) Seleccionar Id, Nome e Idade de todas as pessoas com Idade inferior a 30 anos ou superior a 40 anos

Ou ainda

OUTROS OPERADORES
[BETWEEN, IN, IS, LIKE]

10) Seleccionar o Id, Nome e salário de todas as pessoas com idade entre 30 e 40 anos

11) Seleccionar Id, Nome e Idade de todas as pessoas com Idade inferior a 30 anos ou superior a 40 anos

-- Ou ainda

12) Seleccionar o código postal completo de Lisboa e Tomar

Ou ainda

13) Seleccionar as cidades que não tenham código postal: 1000, 1100, 1500

14) Seleccionar todas as pessoas que não têm telefone

15) Seleccionar todas as pessoas que têm telefone

16) Seleccionar todas as mensagens que contenham a palavra vendas

17) Seleccionar todos os nomes de pessoas que contenham a string 'da'

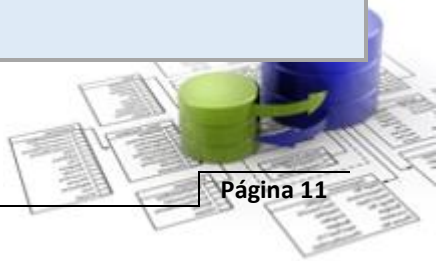
ORDENAÇÃO

Como proceder à ordenação do resultado produzido por uma query SQL?

COMANDO ORDER BY

```
SELECT Campo1, Campo2, ..., *
FROM Tabela1, Tabela2, ..., TabelaK
[WHERE CONDIÇÃO ...]
[GROUP BY...]
[HAVING ...]
[ORDER BY Campo [ASC|DESC], Campo [ASC|DESC], ...];
```

OSB
ASC = Ascendente
DESC = Descendente
Por defeito a ordenação é ascendente



18) Seleccionar todas as pessoas e ordená-las por ordem ascendente da sua idade

É equivalente, por defeito, a:

19) Seleccionar o Nome e Salario de todas as Pessoas e produzir o resultado ordenado por forma a que os maiores salários figurem no topo.

20) Seleccionar o Nome e Salario de todas as Pessoas, que tenham telefone, e produzir o resultado ordenado por forma a que os maiores salários figurem no topo.

ORDENAÇÃO POR VÁRIAS COLUNAS

21) Apresente o conteúdo da tabela Comissão ordenado por ordem ascendente do Id e Id_msg

22) Apresente o conteúdo da tabela Comissão com id<40 ordenado por ordem ascendente do Id e ordem descendente do Valor.

SELECÇÃO DE EXPRESSÕES
O Comando SELECT, para além da selecção de colunas permite, também, a selecção de expressões e constantes

23) Seleccione o nome e idade de todas as pessoas e apresente a idade que terão no próximo ano. Ordene o resultado pelo Nome.

Alteração do Nome das Colunas
O nome com que uma coluna é apresentada no resultado pode ser diferente do nome original.
O comando AS novo_nome a seguir à coluna ou expressão que se pretende alterar muda a designação dessa mesma coluna ou expressão.

24) Exemplo da utilização do comando AS

25) Transpor um valor constante ou Expressão constante para um campo

CÁLCULOS SOBRE VALORES DAS TABELAS

26) Seleccione na tabela comissão o id e o valor ilíquido a receber e, em função deste valor, calcule o montante de imposto a reter a uma taxa 17% e o montante liquido a receber para os indivíduos com numero de id igual a 14 ou 25 anos. Ordene o resultado por ordem descendente do valor líquido.

Eliminação de Linhas repetidas no resultado de uma consulta
CLAUSULA DISTINCT

27) Liste todas as localidades presentes na tabela postal

OBS:
Repare que a localidade Lisboa se repete por várias vezes.
Como proceder para que passe a figurar apenas uma vez?
-> Utilizar o comando DISTINCT

JUNÇÃO DE TABELAS
A operação de JOIN "Junção de tabelas" permite extrair, numa única Query informação contida em diferentes tabelas

```
SELECT Campo1, Campo2, ..., CampoN, *
FROM Tabela1, ..., TabelaK;
```

CROSS JOIN ou PRODUTO CARTESIANO
(Junção Simples)

```
SELECT *
FROM Pessoa, Postal;
```

O Produto Cartesiano vai associar a cada linha da tabela pessoa o conjunto das linhas da tabela Postal

INNER JOIN
(Junção por chaves)
Operação que permite a junção de duas ou mais tabelas, por intermédio do relacionamento Chave-Principal/Chave-Estrangeira.
Apenas são apresentados os registos em que se verifique ligação entre as chaves

```
SELECT Nome, Cod_Postal, Local
FROM Pessoa, Postal
WHERE Cod_Postal = Codigo;
```

28) Seleccionar as localidades onde não existem telefones

Nota:
Se os campos chave tiverem o mesmo nome na tabela MASTER e na Tabela DETAIL, faz-se a distinção usando Nome_Tabela. Nome_Campo

29) Seleccionar todas as comissões a pagar às pessoas

30) Seleccionar o Nome e o Valor das Comissões, ordenando o resultado pelo Nome e maiores valores de comissões.

31) Seleccionar todas as comissões, os seus destinatários, e a descrição das operações que reflectem. O resultado deve ser ordenado pelo Nome.

Nome	Valor	Mensagem
António Dias	170	Fretes Individuais
Isabel Espada	20	Vendas Extra
Isabel Espada	14230	Fretes Empresas
Isabel Espada	5500	Comissão de Vendas
Isabel Espada	120	Deslocações
José António	2300	Fretes Individuais
Nascimento Augusto	10500	Comissão de Vendas
Nascimento Augusto	3750	Ofertas
Nascimento Augusto	400	Combustíveis
Nascimento Augusto	2600	Refeições
Paulo Viegas	2500	Comissão de Vendas
Paulo Viegas	370	Fretes Empresas

NOTA:
A junção de N tabelas num único SELECT obriga a efectuar, pelo menos, n-1 condições de junção
O nome de uma tabela num comando SELECT pode ser alterado por intermédio de um Alias

```
SELECT  Nome, Mensagem
FROM   Pessoa P, Comissao C
WHERE  P.Id=C.Id;
```

32) Construir uma query que permita visualizar o nome da pessoa, mensagem e valor (resultados ordenados pelo nome). Utilizare “aliases”.

OUTER JOIN
Num Join tradicional (INNER JOIN) a junção entre duas tabelas T1 e T2 refere apenas os valores de T1 que têm correspondência em T2.
No OUTER JOIN estende este conceito, ou seja refere também os valores de T1 que não tenham correspondente em T2.
O OUTER JOIN pode ser realizado à esquerda - LEFT JOIN- ou à direita - RIGHT JOIN - , isto é em função da tabela T1 ou T2.

Comparação entre INNER JOIN com OUTER JOIN

INNER JOIN Clássico

```
SELECT Nome, Cod_Postal, Codigo, Local
FROM Postal, Pessoa
WHERE codigo=cod_Postal;
```

Outra variante de INNER JOIN

```
SELECT Nome, Cod_Postal, Codigo, Local
FROM Postal
INNER JOIN Pessoa
  ON Postal.Codigo=Pessoa.Cod_Postal;
```

OUTER JOIN apresenta duas variantes: LEFT JOIN e RIGHT JOIN

LEFT JOIN

```
SELECT Nome, Cod_Postal, Codigo, Local
FROM Postal
LEFT JOIN Pessoa
  ON Postal.codigo=Pessoa.Cod_Postal;

SELECT Nome, Cod_Postal, Codigo, Local
FROM Pessoa
LEFT JOIN Postal
  ON Postal.codigo=Pessoa.Cod_Postal;
```

Nome	Cod_Postal	Codigo	Local
NULL	NULL	1000	LISBOA
Isabel Espada	1100	1100	LISBOA
Célia Morais	1100	1100	LISBOA
NULL	NULL	1200	LISBOA
José António	1500	1500	LISBOA
António Dias	1500	1500	LISBOA
Paulo Viegas	1500	1500	LISBOA
NULL	NULL	2000	SANTAREM
Nascimento Augusto	2300	2300	TOMAR
NULL	NULL	3000	COIMBRA
Florinda Simões	4000	4000	PORTO
NULL	NULL	9000	FUNCHAL

Nome	Cod_Postal	Codigo	Local
Célia Morais	1100	1100	LISBOA
Nascimento Augusto	2300	2300	TOMAR
Paulo Viegas	1500	1500	LISBOA
Florinda Simões	4000	4000	PORTO
Isabel Espada	1100	1100	LISBOA
António Dias	1500	1500	LISBOA
José António	1500	1500	LISBOA

Quando o OUTER JOIN é realizado à esquerda, são considerados todos os registos da tabela da esquerda e apenas os registos correspondentes na tabela da direita.

RIGHT JOIN

```
SELECT Nome, Valor
FROM Comissao
RIGHT JOIN Pessoa
  ON Comissao.id=Pessoa.id;

SELECT Nome, Valor
FROM pessoa
RIGHT JOIN comissao
  ON Comissao.id=Pessoa.id;
```

Nome	Valor
Célia Morais	NULL
Nascimento Augusto	2600
Nascimento Augusto	400
Nascimento Augusto	3750
Nascimento Augusto	10500
Paulo Viegas	370
Paulo Viegas	2500
Florinda Simões	NULL
Isabel Espada	120
Isabel Espada	5500
Isabel Espada	14230
Isabel Espada	20
António Dias	170
José António	2300

Nome	Valor
Nascimento Augusto	10500
Paulo Viegas	2500
Nascimento Augusto	3750
Nascimento Augusto	400
Isabel Espada	20
Isabel Espada	14230
Isabel Espada	5500
Nascimento Augusto	2600
Paulo Viegas	370
NULL	20
Isabel Espada	120
António Dias	170
José António	2300

Quando o OUTER JOIN é realizado à direita são considerados todos os registos da tabela da direita e apenas os registos correspondentes na tabela da esquerda.

Exemplo LEFT e RIGHT JOIN:

```
SELECT Nome, Valor, Local
FROM (
      Pessoa
      LEFT JOIN Postal
        ON Pessoa.Cod_Postal=Postal.Codigo
      )
RIGHT JOIN Comissao
  ON Pessoa.id=Comissao.id;
```

Nome	Valor	Local
Nascimento Augusto	10500	TOMAR
Paulo Viegas	2500	LISBOA
Nascimento Augusto	3750	TOMAR
Nascimento Augusto	400	TOMAR
Isabel Espada	20	LISBOA
Isabel Espada	14230	LISBOA
Isabel Espada	5500	LISBOA
Nascimento Augusto	2600	TOMAR
Paulo Viegas	370	LISBOA
NULL	20	NULL
Isabel Espada	120	LISBOA
António Dias	170	LISBOA
José António	2300	LISBOA

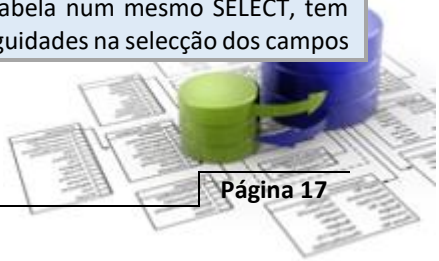
SELF JOIN

O SELF JOIN é uma variante do INNER JOIN que serve para fazer a comparação do conteúdo de duas colunas da mesma tabela.

```
SELECT P1.codigo, P2.local
  FROM Postal P1, Postal P2
WHERE P1.codigo=P2.codigo
      AND P2.local='Lisboa';
```

NOTA:

Quando se coloca duas ou mais vezes a mesma tabela num mesmo SELECT, tem obrigatoriamente que se usar alias para evitar ambiguidades na selecção dos campos



FUNÇÕES DE AGREGAÇÃO OU FUNÇÕES ESTATISTICAS:
MAX "Devolve o valor MAXIMO"
MIN "Devolve o valor MINIMO"
COUNT "Devolve o numero de linhas - CONTADOR"
SUM "Devolve a soma de todos os valores da coluna - SOMATORIO"
AVG "Devolve a média dos valores da coluna - AVERAGE"

33) Contar o número de ocorrências da tabela pessoa

34) Contar o número de pessoas e quantas têm telefone

35) Contar o número de pessoas que não têm telefone

36) Contar o número de comissões com valor superior a 1000

37) Calcular a média das idades das Pessoas

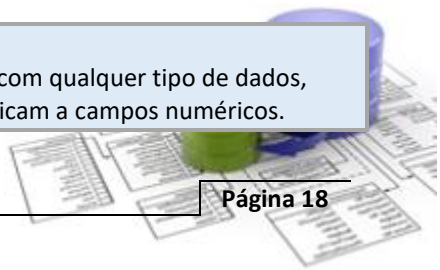
38) Calcular o salário médio das pessoas com mais de 30 anos.

39) Qual o maior valor de Salário

40) Qual a pessoa mais nova

41) Qual o valor total das comissões a pagar?

NOTA:
As funções COUNT, MAX e MIN podem ser usadas com qualquer tipo de dados, enquanto que as funções AVG e SUM apenas se aplicam a campos numéricos.



AGRUPAMENTO DE DADOS

1) Cláusula GROUP BY
2) Cláusula HAVING
3) WHERE vs HAVING

```
SELECT Campo1, ..., CampoN
FROM Tabelas
WHERE (Condição)
GROUP BY ......
HAVING ........
ORDER BY ......
```

1) CLAUSULA GROUP BY
Divide o resultado de um SELECT em grupos de resultados;
Se um comando SELECT contiver uma clausula GROUP BY, então todas as coluna seleccionadas (no SELECT) têm que estar presentes na clausula GROUP BY;

42) Calcular o valor médio de comissões por idade das pessoas

43) Calcular o número de comissões e a média de comissões sobre vendas a receber por cada pessoa, indicando tambem o nome e o local da residência

Nome	local	N_Comissoes	Media_Com
Nascimento Augusto	TOMAR	1	10500.0000
Paulo Viegas	LISBOA	1	2500.0000
Isabel Espada	LISBOA	2	2760.0000

44) Calcule o valor médio das comissões agrupadas por ID "Identificador Comissão"

2) CLAUSULA HAVING
- Serve para impor restrições aos grupos que são processados;
- Quando se pretender mostra grupos de dados que apresentem uma determinada característica em particular deve-se utilizar a cláusula HAVING, que actua unicamente sobre o resultado dos grupos e não a clausula WHERE que actua sobre as linhas da tabela.

45) Vamos pegar no problema anterior, mas mostrando apenas os grupos com valores médios de comissões superiores a 1000.

Id	TOTAL
14	4312.50
25	1435.00
37	4967.50
49	2300.00

46) Apresente o nome de todas as pessoas que possuam mensagens com Id > 20

nome	Id_msg
Isabel Espada	40
António Dias	20
José António	20

3) WHERE vs HAVING
A clausula WHERE utiliza-se para impor restrições sobre os registos da BD.
A clausula HAVING serve para impor restrições aos grupos de registos que foram seleccionados por via da condição WHERE.

ORDENAÇÃO DE GRUPOS
O processo de ordenação sobre grupos segue as mesmas regras da ordenação de registos, isto é, pode fazer-se a ordenação por colunas ou por expressões.

Ordenação por colunas
```
SELECT Id, COUNT (Id)
  FROM Comissao
GROUP BY id
ORDER BY id;
```

Ordenação por expressões
```
SELECT id, COUNT (id)
  FROM Comissao
GROUP BY id
ORDER BY COUNT(Id);
```

SubQUERY
Uma Subquery consiste num comando SELECT dentro de outro comando SELECT.

Existem dois tipos de Subquerys:
Querys Correlacionadas
em que o sentido da execução é de fora para dentro
Querys Não Correlacionadas
em que o sentido da execução é de dentro para fora

Uma SubQuery pode existir:
Dentro de um SELECT,
Dentro de uma subquery
Nos comandos INSERT, UPDATE e DELETE
Na definição de VIEWs

Subquery Não Correlacionada

Listar a pessoa que detém menor salário.
1. Calcular o menor salario,

```
SELECT MIN(salario) AS Menor_Salario
FROM Pessoa
GROUP BY Id;
```

O menor salario é :74000

2. Associar o menor salário ao nome de uma ou varias pessoas

```
SELECT Nome
FROM Pessoa
WHERE Salario=74000;
```

47) Listar a pessoa que detém menor salário.

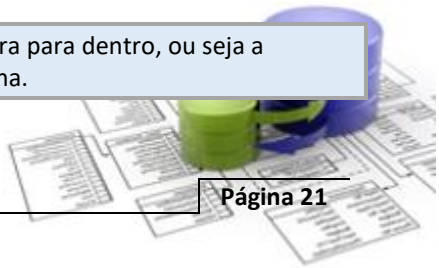
48) Calcular o nome da pessoa com salário > que a media de salários

1- Seleccionar o nome e o salário das pessoas

2- Comparar o salário de cada um com a média dos salários

SubQuery Correlacionada

Neste tipo de Querys o sentido da execução é de fora para dentro, ou seja a execução da query interna depende da query externa.



49) Calcular as pessoas cujo valor de salário é superior ao valor das comissões recebidas.

O sentido da execução desta query é de fora para dentro, pois compete ao SELECT exterior enviar o campo Pessoa.id para que o Select interno possa calcular a soma das comissões dessa mesma pessoa.

OPERADOR IN

O **Operador IN** permite verificar a existencia de um determinado valor, num conjunto de valores colocados entre parêntesis. Este operador verifica a existência de um determinado valor no resultado de uma query, e em função disso devolve TRUE ou FALSE.

50) Calcular o número de pessoas que vivem em Lisboa.

Resulução sem subconsulta

```
SELECT Codigo
FROM Postal
WHERE Local = 'Lisboa';
```

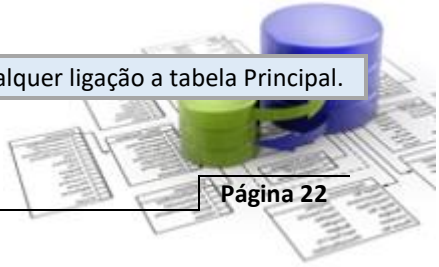
Resultado: 1000, 1100, 1200, 1500

```
SELECT Nome, Cod_Postal
FROM Pessoa
WHERE Cod_postal IN (1000, 1100, 1200, 1500);
```

51) Calcular o conjunto das pessoas que não vive em Lisboa

52) Verificar se existe algum Id na tabela Comissão (Onde figura como chave externa de ligação a tabela pessoa) que não tenha ocorrências na tabela Pessoa.

Resultado: Ficamos a saber que o Id 40 não tem qualquer ligação a tabela Principal.



OPERADOR EXISTS

> • O operador EXISTS serve para avaliar o resultado de subquerys
> • O objectivo é verificar se da execução de uma subquery, resultou algum valor, ou valores de retorno, explicitados sob a forma de TRUE Ou FALSE.
> • Como o operador EXISTS apenas devolve TRUE ou FALSE como resultado, as subquerys associadas por norma são do Tipo SELECT *.
> • O Operador NOT EXISTS devolve TRUE quando não resulta qualquer linha da execução de uma subquery.

53) Seleccione todas as linhas da tabela postal cujo codigo é <= a 5000 e >= a 3000.

Sem sub-consulta

OPERADORES ALL E ANY

> Os operadores ALL e ANY podem ser usados para fazer comparações com base nos operadores (= , >, <, <>, <=, >=)
>
> ANY: retorna verdade se ALGUM dos resultados (pelo menos 1 linha) da subquery satisfazem a condição
> ALL: retorna verdade se TODOS os valores (linhas) da subquery satisfazem a condição

54) Seleccione o nome e a idade de todas as pessoas que não vivem em Lisboa.

Outra Forma

COMANDO INSERT

```
INSERT INTO Nome_da_Tabela (Campo1, ...CampoN)
VALUES (Valor1, ..., ValorN);

Outra forma
INSERT INTO Nome_da_Tabela VALUES (Valor1, ..., ValorN);
```

Exemplos
INSERT INTO Postal (Codigo, Local) VALUES (1250, 'Mouraria');
Ou
INSERT INTO Postal VALUES (1250, 'Mouraria');

55) Inserir os seguintes dados na tabela Pessoa

ID=999

Nome= Anabela

Idade=35

Salario=560000

Telefone= Não Tem

Cod_Postal= 2300

Neste caso temos que optar pela segunda possibilidade uma vez que o indivíduo não tem telefone, o que obriga a especificar os campos.

Cuidados a ter com o Comando INSERT:
- Não inserir mais do que uma vez a mesma chave primária;
- Não inserir mais do que uma vez o mesmo valor num campo Índice
- Não inserir o valor NULL num campo NOT NULL;
- Não inserir dados nas colunas que não sejam compatíveis com o formato das mesmas;
- O número de valores a introduzir tem que ser igual ao número das colunas da tabela.

INSERÇÃO DE CONJUNTOS DE REGISTOS

Imagine que pretende inserir na tabela Postal o Id e o Nome de todas as pessoas

Este comando alterou por completo o conteúdo da tabela Postal.

Vamos repor novamente o conteúdo original

CREATE TABLE

Temos de novo a tabela original.
No entanto devemos ter cuidado com este tipo de operações na medida em que podemos perder o conteúdo da tabela.

COMANDO UPDATE

O Comando UPDATE serve para alterar o conteúdo dos campos de uma tabela.

Sintaxe do comando UPDATE

```
UPDATE Nome_da_Tabela
SET Nome_da_coluna1 = (Expressão1, Query1),
    Nome_da_Coluna_N = (ExpressãoN, QueryN)
WHERE Condição;
```

56) Imagine que quer aumentar o salário de todos os indivíduos em 10%.

57) Imagine que pretende reduzir em 5% o salário do Paulo Viegas e aumentar a idade em 5 anos.

58) Imagine que pretende adicionar o Prefixo 01 aos telefones de Lisboa

Outra Forma

COMANDO DELETE

Sintaxe do Comando
DELETE FROM tabela
WHERE Condição;

Como apagar todas a linhas de uma tabela, ou seja deixar a tabela vazia

OBS: Esta tabela não existe.

```
DELETE
FROM Viatura;
```

59) Apagar todas as localidades, na tabela Postal, que não estejam associadas a ninguém.

1. Vamos verificar quais são as localidades

2. Para apagar estas localidades basta aplicar o mesmo algoritmo com o comando DELETE

Verificar:

NOTAS:
• O Comando DELETE apaga as linhas da tabela, mas esta continua a existir.
• Se na base de dados estiver definida integridade referencial (REFERENCES) entre a tabela A e B, então so podemos apagar uma linha de A quando em B não existirem referencias a essa linha;
• Se estiver definida uma relação do tipo CASCADE DELETE entre as tabelas A e B, e se um determinado campo X for Chave Primária em A e Chave Estrangeira em B, então sempre que apagar uma linha de A, automaticamente serão eliminadas todas as linhas de B que lhe estiverem associadas.

MANIPULAÇÃO DE DATAS

O Formatos das datas pode variar em diferentes SGBD.
No MySql o formato é:
Data ´aaaa-mm-dd'
Hora ´hh:mm´

60) Criar uma nova tabela temporária com os seguintes campos:

A Date, T Time

VIEWS e INDICES

COMANDO CREATE VIEW

Noção de VIEW
• Funciona como uma janela para a Base de Dados
• Corresponde à criação de uma tabela Virtual

Qual a finalidade das VIEWS
• Segurança: Restringe os campos ou as linhas a disponibilizar para cada um dos perfis de utilizador da Base Dados
• Confidencialidade: Evita-se que determinados utilizadores tenham acesso a determinado tipos de dados.
• Simplicidade: Numa única VIEW podemos condensar o acesso a várias tabelas.

```
CREATE VIEW Nome_da_View (Coluna 1, ... ColunaN)
   AS SELECT (Condição);
```

61) Criar uma View para o conjunto de pessoas cujo salário é menor que 100000.

Verificar:

62) Crie uma View para o conjunto das pessoas cujo salário é menor que 100000, e paras os quais se apresentem apenas os campos Id, Nome e Salário.

Verificar:

63) Crie um VIEW com as características da VIEW anterior mas altere o nome dos campos para Num, Nome, Ordenado.

Verificar:

64) Altere o salário da Isabel Espada para 92000, mas fazendo uso da VIEW People02.

Confirmar que o valor foi alterado na tabela original:

65) Crie uma VIEW Moradas contendo o nome, codigo postal e localidade de todas as pessoas.

Verificar:

66) Crie uma VIEW Ordenado que apresente o salário e o Total de Comissões a receber por cada pessoa e apenas para as pessoas que tenham comissões a receber.

> Nota:
> Neste caso temos que usar a cláusula GROUP BY para agrupar as varias comissões que um individuo pode ter a receber.

67) Crie uma VIEW Resultado que apresente o conjunto das pessoas que ganham menos de 100000 e que têm comissões a receber.

Nota:
Neste caso podemos criar a VIEW resultado com base na VIEW People02 (Mostra pessoas que ganham menos 100000) e a VIEW Ordenado (Mostra o valor das comissões por pessoa.

COMANDO DROP VIEW

Permite eliminar uma VIEW da Base de Dados

INDEXAÇÃO (Índices)

1. Indexação é um processo usado para facilitar o acesso aos registos de uma determinada tabela.
2. Este processo é mais premente para tabelas com muitos registos e sujeitas a muitas pesquisas.
3. O Processo de indexação apoia-se no conceito de ARVORE BINARIA
4. Os nos da Arvore Binária guardam sempre dois valores:
   - O Valor da chave de indexação (Campo Index)
   - A posição (física) do registo na respectiva tabela.
5. Contrariamente as VIEWs que eram estruturas virtuais, as Arvores Binárias existem fisicamente no disco.

COMANDO CREATE INDEX

68) Criar um índice sobre o campo Local da tabela Postal.

Neste caso vamos admitir que a tabela postal tem muitas localidades e que o acesso mais solicitado é por intermédio do nome das mesmas

69) Criar um índice descendente tendo por base o valor dos salários

70) Criar um índice único para o nome das pessoas.

Neste caso é que para cada registo é criado um índice ao invés dos anteriores que criam um índice para um conjunto de registos.

Listas índices

Para listar todos os índices de uma tabela especifica:
SHOW INDEX FROM table_name FROM db_name;

Ou:
SHOW INDEX FROM db_name.table_name;

A chave primária é um índice: Índices PRIMARY KEY

71) Visualizar todos os índices da tabela pessoa.

```
SHOW INDEX FROM pessoa FROM ex1_caderno;

Ou
SHOW INDEX FROM pessoa;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
pessoa	0	PRIMARY	1	Id	A	7	NULL	NULL		BTREE			YES	NULL
pessoa	1	iSalario	1	Salario	D	7	NULL	NULL		BTREE			YES	NULL

72) Visualizar todos os índices da tabela postal.

```
SHOW INDEX FROM ex1_caderno.postal;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
postal	0	PRIMARY	1	Codigo	A	9	NULL	NULL		BTREE			YES	NULL
postal	1	iLocal	1	Local	A	6	NULL	NULL		BTREE			YES	NULL

73) Visualizar todos os índices da BD

COMANDO DROP INDEX

74) Apagar o índice indNome

TRANSACÇÕES

1. Uma transacção consiste numa unidade lógica de trabalho. Se a transacção envolver mais do que uma tarefa (Processo) então ou se realizam todos ou não se realiza nenhum.
2. Os SGBD fornecem um mecanismo denominado Gestão Transaccional, cujo objectivo é proceder ao controlo das transacções.
3. Alguns SGBDR têm ativa, por defito a opção AUTO COMMIT.

Desativar autocommit:

```
SET autocommit=0;
Ou
SET autocommit = OFF
```

Ativar:

```
SET autocommit = 1;
Ou
```

```
SET autocommit = ON;
```

75) Imagine que numa determinada operação pretende transferir 100 Euros da conta do Sr. A para a conta do Sr. B.

Este processo envolve 2 transacções:

1.

2.

76) Imagine que no final da primeira transacção havia uma falha no sistema. O Processo fica incompleto e provavelmente íamos ter problemas com o Sr. A

COMMIT / ROLLBACK / SAVEPOINT / START TRANSACTION
COMMIT: Sempre que se digita COMMIT as transacções são transferidas fisicamente para a Base de Dados e um novo processo se inicia.
ROLLBACK: Anula todas as operações efectuadas ate ao comando COMMIT anterior
SAVEPOINT: Marca um determinado ponto no processo. Neste caso só é possível fazer ROLLBACK até o comando SAVEPOINT.
O Uso de SAVEPOINT's não invalida que as transacções marcadas só se realizem no final, com o uso de COMMIT.

```
SAVEPOINT identifier
ROLLBACK [WORK] TO [SAVEPOINT] identifier
RELEASE SAVEPOINT identifier
```

77) Exemplo

```
+------+--------+
| a    | b      |
+------+--------+
|   10 | Heikki |
+------+--------+
```

SEGURANÇA

Os mecanismos de segurança assentam num conjunto de permissões, que podem efectuar-se de uma forma lógica, por via das VIEWs ou fisicamente com configuração de diferentes perfis de utilizador.

Criação de um utilizador

Sintaxe
CREATE USER Nome_Utilizador IDENTIFIED BY 'Password';

78) Exemplo: Criar o utilizador “joana” com a password “123456789”.

Listar:

79) Mudar a password do utilizador “joana” para “987654321”

SET PASSWORD FOR ‘joana’@’localhost’ = PASSWORD (‘987654321’);

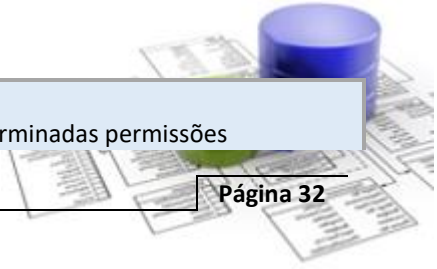
80) Apagar utilizador “joana”

DROP USER ‘joana’@’localhost’

COMANDO GRANT / REVOKE

GRANT

GRANT
Comando usado para proceder a atribuição de determinadas permissões



Sintaxe
Dar permissões para todas a s tabelas da BD

```
GRANT <previlegio>
ON <nome_BD>.*
To <Nome_Utilizador>;
```

81) Dar permissões para todas as Bases de dados

82) Dar acesso apenas para determinadas tabelas de uma base de dados

REVOKE

Comando utilizado para eliminar permissões.

```
REVOKE <previlegio>
On <Nome_BD>
To <Nome_Utilizador>;
```

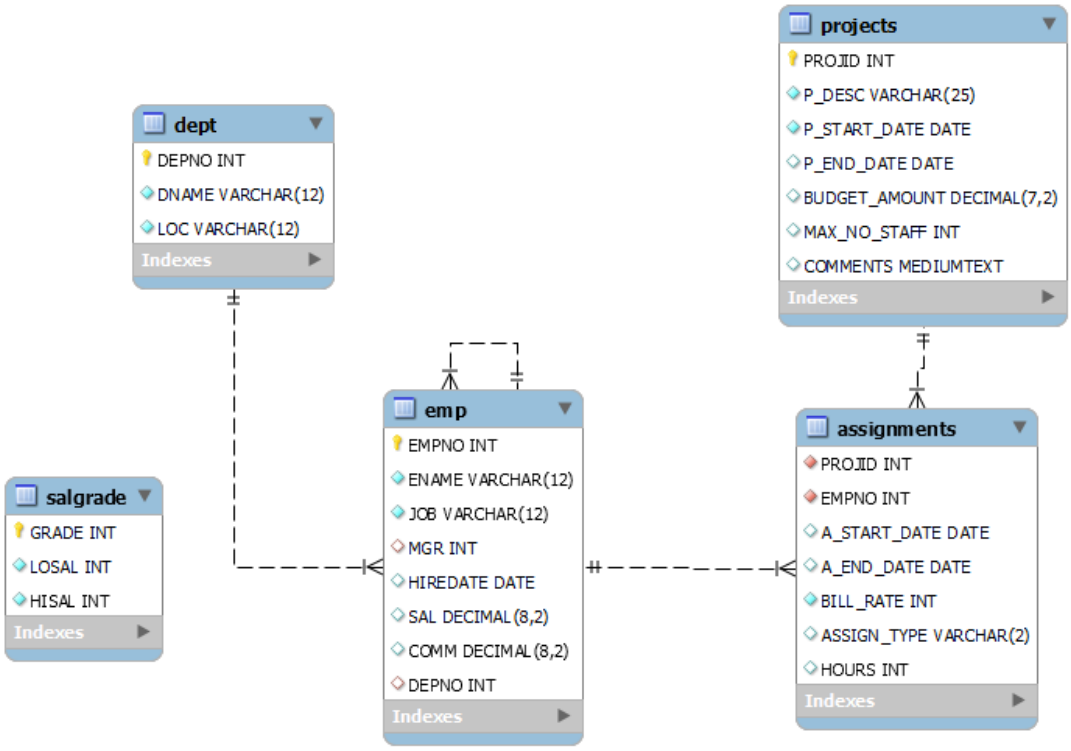
Eliminar todas as permissões ao utilizador “joana”

Exercício 4 | SQL / base de dados "Gestão de Projetos"

1) Com base no SQL seguinte, desenhar o diagrama relacional.

2) No MySQL, criar a base de dados "gestao_projetos" e respectivas tabelas/restrições/relações/… SQL e registos disponíveis no virtual.

a) Modelo relacional



b) Dicionário de dados

Tabela EMP

Nome coluna	Tipo Dados	Tamanho	Chaves	Restrições
Empno (Num. Empregado)	NUMBER	4	Pr_Key	Primary Key
Ename (Nome Empregado)	VARCHAR2	12		Not Null
Job (Profissão)	VARCHAR2	12		Not Null
Mgr (Chefe)	NUMBER	4		Emp_mgr Emp (empno)
Hiredate (Data admissão)	DATE			Default Sysdate
Sal (Salário)	NUMBER	8,2		Between 500 – 6000
Comm (Comissões)	NUMBER	8,2		
Depno (Num. Departamento)	NUMBER	2	Fr_Key	Dept (depno)

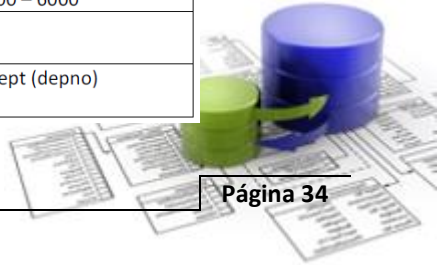


Tabela DEPT

Nome coluna	Tipo Dados	Tamanho	Chaves	Restrições
Depno (Num. Departamento)	NUMBER	2	Pr_Key	Primary Key
Dname (Nome departamento)	VARCHAR2	12		Not Null
loc (Localidade)	VARCHAR2	12		Not Null

Tabela SALGRADE

Nome coluna	Tipo Dados	Tamanho	Chaves	Restrições
Grade (Nível remuneração)	NUMBER	2	Pr_Key	Primary Key
Losal (Salário Mínimo)	NUMBER	8		Not Null
Hisal (Salário Máximo)	NUMBER	8		Not Null

Tabela ASSIGNMENTS

Nome coluna	Tipo Dados	Tamanho	Chaves	Restrições
Projid (Num. Projecto)	NUMBER	4	Fr_Key	Not Null Projects (projid)
Empno (Num. Empregado)	NUMBER	4	Fr_Key	Not Null Emp (empno)
A_start_date (Data inicio tarefa)	DATE			Not Null
A_end_date (Data final tarefa)	DATE			>= A_start_date
Bill_rate (Taxa Execução)	NUMBER	4,2		Not Null
Assign_Type (Tipo Tarefa)	VARCHAR2	2		
Hours (Horas Trabalho)	NUMBER	3		

Tabela PROJECTS

Nome coluna	Tipo Dados	Tamanho	Chaves	Restrições
Projid (Num. Projecto)	NUMBER	4	Pr_Key	Primary Key
P_desc (Descrição projecto)	VARCHAR2	30		Not Null
P_Start_date (Data inicio projecto)	DATE			Not Null
P_end_date (Data conclusão projecto)	DATE	4		>= P_start_date
Budget_amount (Valor gasto)	NUMBER	7,2		
Max_no_staff (Membros do staff)	NUMBER	2		

c) Registos

Tabela EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPNO
7369	SMITH	CLERK	7902	12/17/1980	800	-	20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7566	JONES	MANAGER	7839	04/02/1981	2975	-	20
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	05/01/1981	2850	-	30
7782	CLARK	MANAGER	7839	06/09/1981	2450	-	10
7788	SCOTT	ANALYST	7566	04/19/1987	3000	-	20
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	05/23/1987	1100	-	20
7900	JAMES	CLERK	7698	12/03/1981	950	-	30
7902	FORD	ANALYST	7566	12/03/1981	3000	-	20
7934	MILLER	CLERK	7782	01/23/1982	1300	-	10

Tabela DEPT

depno	dname	location
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Tabela SALGRADE

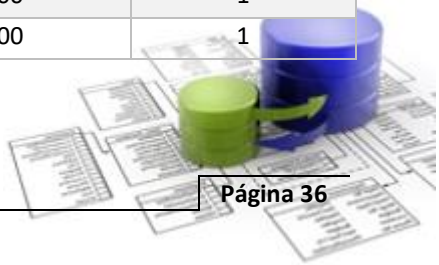
grade	losal	hisal
1	700.00	1200.00
2	1201.00	1400.00
4	2001.00	3000.00
5	3001.00	99999.00
3	1401.00	2000.00

Tabela ASSIGNMENTS

PROJID	EMPNO	A_START_DATE	A_END_DATE	BILL_RATE	ASSIGN_TYPE	HOURS
1	7369	01-JAN-88	03-JAN-88	50	WR	15
1	7902	04-JAN-88	07-JAN-88	55	WR	20
2	7844	01-JAN-89	10-JAN-89	45,5	PF	30

Tabela PROJECTS

PROJID	P_DESC	P_START_DATE	P_END_DATE	BUDGET_AMOUNT	MAX_NO_STAFF
1	WRITE CO30 COURSE	02-JAN-88	07-JAN-89	500	1
2	PROOF READ NOTES	01-JAN-89	10-JAN-89	600	1



3) Criar as seguintes tabelas e restrições

4) Criar as seguintes FOREIGN KEY

```
EMP                 DEPNO_FR_KEY
      DEPT

EMP                 EMP_MGR FOREIGN
      EMP

ASSIGNMENTS
```

PROJECTS

ASSIGNMENTS
EMP •

5) Verificar as restrições.

6) Acrescentar uma coluna na tabela EMP

Verificar:

7) Eliminar a adicionada anteriormente à tabela EMP

Verificar:

8) Acrescentar uma restrição na tabela EMP

9) Alterar o tamanho da coluna ename (tabela EMP) para 25 caracteres

Verificar:

10) Insira os seguintes dados nas tabelas criadas anteriormente.

a) Tabela DEPT

Verificar se os dados foram corretamente inseridos:

a) Tabela EMP

Verificar se os dados foram corretamente inseridos:

a) Tabela SALGRADE

Verificar se os dados foram corretamente inseridos:

a) Tabela ASSIGNMENTS

Verificar se os dados foram corretamente inseridos:

a) Tabela PROJECTS

Verificar se os dados foram corretamente inseridos:

11) Alterar o registo do funcionário 'SCOTT' na tabela EMP para:

- job= 'Salesman',
- Hiredate = '2022-01-01',
- Sal = aumento do salário de 10%

Verificar se os dados foram corretamente alterados:

12) Selecione toda a informação da tabela Salgrade.

13) Liste todos os empregados que têm um salário entre 1000 e 2000.

14) Liste os Números e nomes dos departamentos ordenados pelos respetivos nomes.

15) Apresente os diferentes tipos de profissões.

16) Liste a informação detalhada dos empregados dos departamentos 10 e 20 por ordem alfabética do nome.

17) Liste os nomes e as funções de todos os empregados de escritório (Clerk) do departamento 20.

18) Apresente todos os nomes dos empregados que incluam TH ou LL.

19) Apresente os dados de todos os empregados que têm chefe.

20) Apresente o nome e a remuneração total (12 vezes o salário mais a comissão) para todos os empregados.

21) Apresente todos os empregados que foram admitidos em 1981.

	EMPNO	ENAME	HIREDATE
1	7839	KING	84.07.09
2	7698	BLAKE	84.06.11
3	7521	WARD	84.03.26
4	7782	CLARK	84.05.14
5	7788	SCOTT	84.03.05
6	7844	TURNER	84.07.04
7	7876	ADAMS	84.07.04
8	7900	JAMES	84.07.23

22) Apresente o nome, salário anual (12 * Sal) e comissões de todo o pessoal de vendas (Salesman) cujo salário mensal seja maior que a comissão. O resultado deve ser apresentado por ordem descendente do salário. Se dois ou mais empregados tiverem o mesmo salário, então ordene-os pelo nome.

23) Liste o nome de empregado e respetivo salário, acrescido de um aumento de 15%, expresso sob a forma de número inteiro.

24) Produza uma listagem, como resultado de uma pesquisa insensível a maiúsculas ou minúsculas, com o nome dos empregados cuja função seja 'clerk'.

25) Calcular a média dos salários de todos os empregados.

OBS: Repare que todas as linhas da tabela EMP são tratadas como um grupo. Uma função de grupo pode ser aplicada a um sub-conjunto de linhas duma tabela utilizando a cláusula WHERE.

26) Encontrar o salário mais baixo recebido pelos empregados de escritório (CLERK).

27) Contar o número de empregados no departamento 20.

28) Escrever uma consulta que permita calcular o salário médio para cada função.

Excluir Linhas ao Utilizar GROUP BY

Podem-se excluir, previamente, linhas com uma cláusula WHERE, antes de as dividir em grupos.

29) Apresentar o salário médio de cada função (JOB) excluindo a função “MANAGER”.

30) Visualizar o salário médio mensal para cada função dentro de cada departamento.

A cláusula GROUP BY pode ainda ser utilizada para fornecer resultados para grupos contidos em grupos.

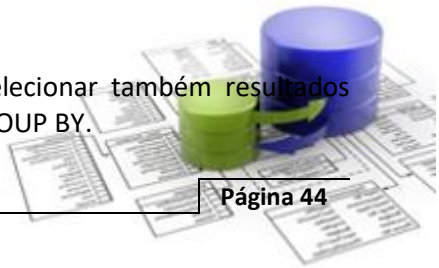
31) Visualizar o salário mais alto para cada grupo de função.

A visualização de JOB é opcional. No entanto, os valores apresentados terão mais significado se estiverem relacionados com um grupo de função.

Esta instrução SELECT visualizará uma lista com mais significado.

Tenha presente a seguinte regra ao utilizar as funções de grupo:

- Se incluir funções de grupo num comando SELECT, não poderá selecionar também resultados individuais A NÃO SER QUE a coluna individual apareça na cláusula GROUP BY.



- Por outras palavras, as funções não podem ser extraídas com itens individuais que não aparecem na cláusula GROUP BY.
- Por exemplo:

ERROR: ORA-0937: not a single group set function

O comando não é válido porque DEPNO tem um valor para cada linha da tabela, enquanto que MIN(SAL) tem um único valor para toda a tabela. Para corrigir o erro, os itens individuais têm que ser agrupados:

No exemplo acima, DEPNO já não é um item individual; é um nome de um grupo. Se incluir mais do que uma coluna com valores individuais na linha de SELECT, terá que agrupar (GROUP BY) todas as colunas individuais. Deste modo, existe uma regra a seguir ao utilizar a cláusula GROUP BY:

- Qualquer coluna ou expressão na lista de SELECT que não seja uma função de grupo terá que ser incluída na cláusula GROUP BY.

32) Apresentar o salário médio de todos os departamentos com mais de três pessoas.

Utilizar a cláusula HAVING se pretender especificar quais os grupos que devem ser visualizados, isto é, restrinja os grupos que serão devolvidos (com base na informação de grupo).

33) Apresentar apenas as funções cujo salário mais alto é igual ou superior a 3000.

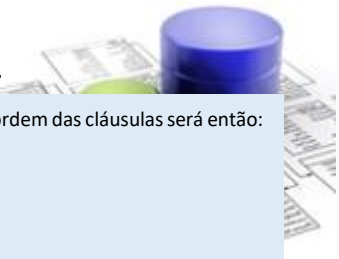
A cláusula HAVING pode preceder a cláusula GROUP BY, mas é recomendado que seja primeiro indicado GROUP BY, por ser mais lógico. Os grupos são formados e são calculadas as funções de grupo antes da cláusula HAVING ser aplicada para selecionar os grupos para o resultado.

34) A cláusula WHERE não pode ser utilizada para restringir os grupos que vão ser devolvidos. A clausula WHERE é ilegal na seguinte instrução de SQL.

```
SELECT DEPNO,AVG(SAL)
FROM EMP
WHERE AVG(SAL) >2000
GROUP BY DEPNO;
```

ERROR: ORA-0934: set function is not allowed here
Só podemos utilizar WHERE para restringir linhas isoladas. Para excluir ou incluir grupos, utilize a cláusula HAVING.

35) Excluir todos os 'MANAGER' utilizando uma cláusula WHERE ao agregar por função.



A cláusula WHERE pode ainda ser utilizada para excluir linhas. A ordem das cláusulas será então:

```
SELECT coluna (s)
FROM tabela(s)
WHERE condições de linha
GROUP BY coluna(s)
HAVING condições de grupo de linhas
ORDER BY coluna(s);
```

36) Calcule o salário mais baixo de todos os empregados.

37) Calcule o salário mais baixo, mais alto e médio de todos os empregados.

38) Liste os salários mais baixos e mais alto de cada tipo de função.

39) Descubra quantos "managers" existem, sem os listar.

40) Encontre o salário médio e a remuneração total média (12 vezes o salário mais a comissão) para cada tipo de função. Não esqueça que os vendedores ganham comissão.

41) Encontre a diferença entre o salário mais alto e o mais baixo.

42) Encontre todos os departamentos que têm mais de três empregados.

43) Verifique se todos os números de empregado são realmente únicos.

44) Liste o salário mais baixo para cada "manager". Exclua os grupos em que o salário mais baixo é inferior a 1000. Ordene o resultado pelo salário.

45) Apresente todos os nomes de empregado e respetivo nome de departamento, por ordem do nome de departamento.

46) Apresente o nome, número e nome de departamento de todos os empregados.

47) Apresente o nome, localidade e departamento dos empregados cujo salário mensal seja superior a 1500.

48) Produza uma lista que mostre o nível salarial de cada empregado.

49) Mostre apenas os empregados com nível 3.

50) Mostre todos os empregados de Dallas.

51) Liste nome, função, salário, nível e nome de departamento para todos os empregados, exceto empregados de escritório (CLERKS). Ordene por salário apresentando primeiro o mais elevado.

52) Liste os seguintes detalhes para os empregados que ganhem 36000 por ano (12 vezes o salário mais a comissão) ou que sejam empregados de escritório (CLERK).

ENAME	JOB	ANNUAL_SAL	DEPNO	DNAME	GRADE
FORD	ANALYST	36000	20	RESEARCH	4
SCOTT	ANALYST	36000	20	RESEARCH	4
MILLER	CLERK	27600	10	ACCOUNTING	4
ADAMS	CLERK	13200	20	RESEARCH	1
JAMES	CLERK	11400	30	SALES	1
SMITH	CLERK	9600	20	RESEARCH	1

6 rows returned in 0.01 seconds Download

53) Consulta para apresentar todos os empregados que ganham menos que os seus managers.

Repare que a cláusula FROM refere EMP duas vezes e, por isso, se atribui um pseudónimo a EMP em ambos os casos - E e M.

A cláusula de junção diz: "onde (where) o número do manager do empregado seja o mesmo que o número de empregado do respetivo manager".

54) Consulta para apresentar todos os empregados e os respetivos managers.

emp_chefe
SMITH works for FORD
ALLEN works for BLAKE
WARD works for BLAKE
JONES works for KING
MARTIN works for BLAKE
BLAKE works for KING
CLARK works for KING
SCOTT works for JONES

55) Crie uma consulta que exibirá o nome dos funcionários, o número do departamento e todos os funcionários que trabalham no mesmo departamento de um determinado funcionário. Forneça a cada coluna um label apropriado.

DEPARTMENT	EMPLOYEE	COLEGA
10	CLARK	KING
10	CLARK	MILLER
10	KING	CLARK
10	KING	MILLER
10	MILLER	CLARK
10	MILLER	KING
20	ADAMS	FORD
20	ADAMS	JONES
20	ADAMS	SCOTT
20	ADAMS	SMITH
20	FORD	ADAMS
20	FORD	JONES
20	FORD	SCOTT
20	FORD	SMITH
20	JONES	ADAMS

56) Crie uma consulta para exibir o nome e a data de admissão de qualquer funcionário admitido após o funcionário Blake.

ENAME	HIREDATE
KING	11/17/1981
CLARK	06/09/1981
SCOTT	04/19/1987
FORD	12/03/1981
MARTIN	09/28/1981
TURNER	09/08/1981
ADAMS	05/23/1987
JAMES	12/03/1981
MILLER	01/23/1982

57) Apresente o departamento que não tem empregados.

58) Liste os empregados por nome e número juntamente com o nome e número do seu manager.

59) Modifique a solução anterior de forma a visualizar KING, que não tem manager.

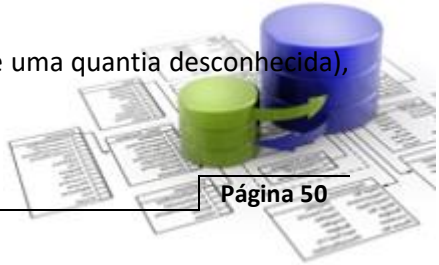
60) Encontre todos os empregados que entraram para a empresa antes do seu manager.

61) Consulta para identificar o empregado (ENAME), função (JOB) com o salário (SAL) mais baixo.

Utilizar subquery

Resolução SEM subquery

Para encontrar o empregado que ganha o salário mais baixo na empresa (é uma quantia desconhecida), são necessários dois passos:



a) Encontrar o salário mais baixo.

MIN(SAL)
800

b) Encontrar o empregado que ganha o salário mais baixo. Neste caso utilizamos o valor 800 que foi devolvido na consulta anterior.

Resolução COM subquery

ENAME	JOB	SAL
SMITH	CLERK	800

62) Escrever uma consulta para encontrar todos os empregados que tenham a mesma função (JOB) que o empregado com o nome "BLAKE".

63) Consulta para encontrar os empregados que ganham o salário mais baixo, em cada departamento.

Sub-consultas de várias linha

O resultado obtido não mostra os departamentos em que trabalham os empregados que se qualificaram.
Além disso, uma vez que estamos apenas a comparar os valores do salário, a consulta interna poderia produzir um valor só porque correspondia ao menor salário de um dos departamentos, não necessariamente o departamento do empregado. Assim, a consulta deveria ser reescrita de modo a fazer corresponder a combinação do salário do empregado e do número do departamento à do salário mínimo e o número do departamento:

ENAME	SAL	DEPNO
SMITH	800	20
JAMES	950	30
MILLER	1300	10

64) Consulta para encontrar os empregados que ganham o salário mais baixo, em cada departamento (versão melhorada).

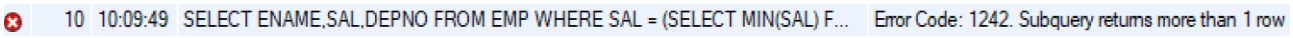
A consulta seguinte encontra de facto os empregados que ganham o menor salário do respetivo departamento - Compara um par de colunas.

ENAME	SAL	DEPNO
JAMES	950	30
SMITH	800	20
MILLER	1300	10

As colunas à ESQUERDA da condição de pesquisa estão entre parêntesis e cada coluna está separada por uma vírgula.
As colunas listadas na cláusula SELECT da subconsulta têm de estar na mesma ordem que a lista de colunas entre parêntesis na cláusula WHERE da consulta externa.
As colunas devolvidas pela consulta interna terão ainda de corresponder em número e em tipo às colunas com que são comparadas na consulta externa.

65) Identifique o erro da seguinte consulta.

Quando uma subconsulta devolve mais do que uma linha e é utilizado um operador de comparação de linha única, o SQL*Plus apresenta a seguinte mensagem de erro:



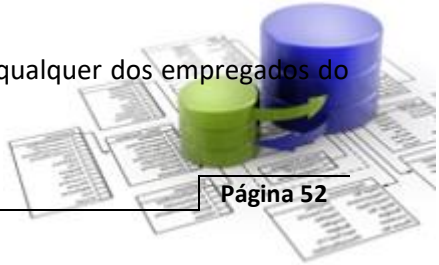
66) Identifique o erro da seguinte consulta.

Neste caso, a consulta interna não devolve nenhuma linha, e obterá: no data found

67) Visualizar os empregados que ganham mais do que o salário mais baixo do Departamento 30.

Ao utilizar SOME/ANY, a palavra-chave DISTINCT é frequentemente utilizada para evitar que as linhas sejam selecionadas mais do que uma vez.

68) Consulta para encontrar todos os empregados que ganham mais do que qualquer dos empregados do Departamento 30.



ALL compara um valor com qualquer dos valores devolvidos por uma subconsulta.

69) Visualizar, os departamentos que tenham um salário médio superior ao do Departamento 30.

CLÁUSULA HAVING COM SUBCONSULTAS ENCADEADAS
As subconsultas encadeadas podem também ser utilizadas na cláusula HAVING.
(Recorde que WHERE se refere a linhas individuais e HAVING a grupos de linhas especificados na cláusula GROUP BY).

70) Construir uma consulta que encontre a função com o salário médio mais elevado

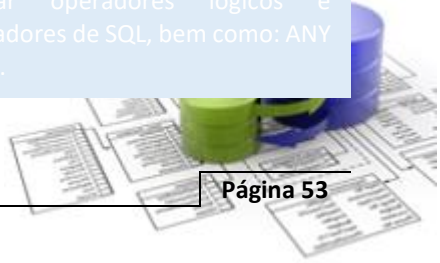
ORDENAÇÃO DE RESULTADOS COM SUBCONSULTAS
Não se pode utilizar a cláusula ORDER BY numa sub consulta.
Mantém-se a regra que só se pode ter uma única cláusula ORDER BY para uma instrução SELECT e, se especificada, terá que ser a última cláusula no comando SELECT.

71) Visualizar o nome, função e data de admissão dos empregados cujo salário é superior ao salário mais elevado de qualquer dos departamentos de SALES.

- A consulta interna deverá ser colocada entre parêntesis, e terá que estar no lado direito da condição.
- A subconsulta não pode conter uma cláusula ORDER BY.
- A cláusula ORDER BY aparece no fim da instrução SELECT principal.
- Colunas múltiplas, na lista de SELECT da consulta interna, terão que estar pela mesma ordem que as colunas que apareçam na cláusula de condição da consulta principal.
O tipo de dados e o número de colunas tem também de corresponder.
- Nas subconsultas podem-se utilizar operadores lógicos e operadores de SQL, bem como: ANY e ALL.

AS SUB-CONSULTAS PODEM:

- Devolver uma ou mais linhas
- Devolver uma ou mais colunas
- Utilizar GROUP BY ou funções de grupo



- Ser utilizadas em vários predicados AND e OR da mesma consulta externa
- Juntar Tabelas
- Extrair dados de uma tabela diferente da tabela da consulta externa

72) Construir uma subconsulta correlacionada para encontrar os empregados que ganhem um salário superior ao salário médio do respetivo departamento:

Passos para a execução de uma subconsulta correlacionada:
1. Obter uma linha candidata (devolvida pela consulta externa).
2. Executar a consulta interna utilizando o valor da linha candidata.
3. Utilizar o valor ou valores resultantes da consulta interna para qualificar ou desqualificar o candidato.
4. Repetir até não restarem linhas candidatas.

Uma subconsulta correlacionada é executada repetidamente, para cada linha candidata da consulta principal.

A Consulta Principal	A Consulta Interna
1. Seleciona a primeira linha candidata - Smith do departamento 20 e que ganha 800. 2. EMP tem na cláusula FROM o pseudónimo E que qualifica a coluna DEPNO referida na cláusula WHERE da consulta interna. 3. A cláusula WHERE compara 800 com o valor devolvido pela consulta interna.	4. Calcula AVG(SAL) para o departamento do empregado. 5. O valor do departamento em WHERE é o departamento do candidato (E.DEPNO) - valor passado à consulta interna a partir da coluna DEPNO da consulta externa. 6. AVG(SAL) para o departamento de Smith - 20 - é 2175. 7. A linha candidata não satisfaz a condição e, por isso, é excluída. 8. Repetir desde o passo 1 para a linha candidata seguinte; ALLEN está no departamento 30 e ganha 1600.

73) Encontrar os empregados que tenham pelo menos um subordinado.

OPERADORES
Quando estiver a encadear instruções de SELECT, todos os operadores lógicos são válidos, bem como ANY e ALL. Além disso, pode-se utilizar o operador EXISTS.
OPERADOR EXISTS
O operador EXISTS é frequentemente utilizado com sub-consultas correlacionadas.
Permite verificar se existe um valor para uma determinada condição.
NOT EXISTS assegura que não existem valores.
Se o valor existir, devolve TRUE (verdadeiro);
Se não existir, assinala FALSE (falso).

74) Encontre todos os empregados cujo departamento não esteja na tabela DEPT.

a) Outro processo de encontrar o departamento que não possui empregados.

Porquê utilizar uma Subconsulta Correlacionada?

A subconsulta correlacionada representa um modo de 'ler' cada uma das linhas da tabela e comparar os valores com determinados dados.

Utiliza-se sempre que uma subconsulta tem de produzir um resultado ou conjunto de resultados diferentes para cada linha candidata considerada pela consulta principal.

O SELECT interno é normalmente executado uma vez por cada linha candidata.

Considerações sobre Eficácia

Agora que foram examinados os dois tipos de subconsultas, vale a pena mencionar que a subconsulta correlacionada (com EXISTS) pode ser o método mais eficaz de efetuar algumas consultas.

O desempenho depende do número de índices, do número de linhas devolvidas pelas consultas, da dimensão das tabelas e do facto de serem ou não necessárias tabelas temporárias para avaliar resultados intermédios. As tabelas temporárias geradas pelo ORACLE não são indexadas, podendo daí resultar urna quebra de desempenho das subconsultas que utilizem IN, ANY e ALL.

NOT EXISTS Versus NOT IN

Embora uma operação NOT IN possa ser tão eficaz quanto NOT EXISTS numa subconsulta, NOT EXISTS é mais fiável se a subconsulta devolver valores NULL, uma vez que uma condição NOT IN é considerada falsa quando se incluem NULLs na lista de comparação.

75) Consulta para imprimir os empregados que não são manager de ninguém.

Nenhuma linha é devolvida pela consulta anterior, uma vez que a coluna MGR contém um valor NULL.

a) A consulta correta é:

76) Encontre os empregados que ganham o salário mais alto em cada tipo de função (JOB).

77) Encontre os empregados que ganham o salário mais baixo em cada função. Visualize o resultado por ordem crescente de salário.

78) Encontre os empregados mais recentes em cada departamento. Ordene por data de admissão.

79) Mostre os detalhes seguintes para cada empregado que ganhe um salário maior que a média do respetivo departamento. Ordene por número de departamento.

80) Liste todos os departamentos em que não existem empregados (utilizando desta vez uma subconsulta).

81) Em que ano entraram mais pessoas para a empresa? Visualize o ano e número de empregados.

82) Modifique a questão 4 para apresentar também o valor do salário médio para o departamento.

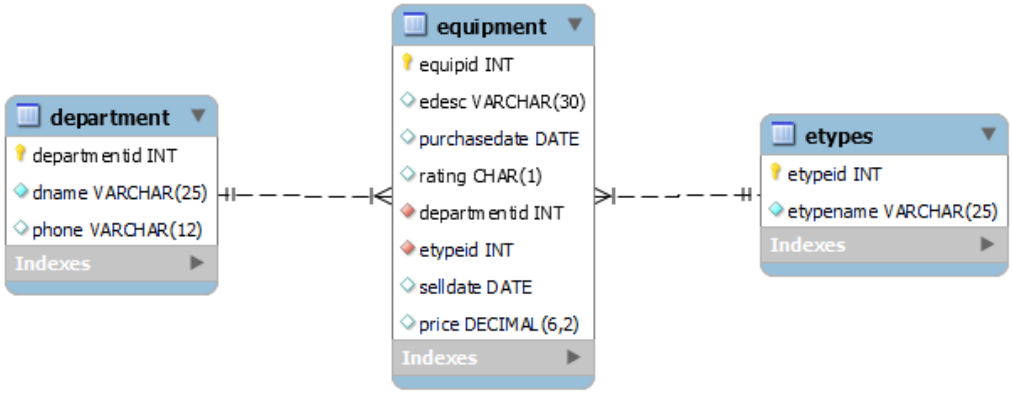
83) Criar uma view que permita visualizar os empregados com o tabalho (job) “SALESMAN”.

84) Listar todos os “SALESMAN” cujo salário é superior a 1400 (executar um select sobre a view “emp_salesman”)

Exercício 5 | SQL / base de dados "Departamentos"

1) Crie a seguinte estrutura, tendo em atenção as PK, FK e restrições:

a) DER



b) Restrições:

- O nome de cada departamento tem que ser único;
- O nome do departamento é obrigatório;
- Cada nome de equipamento (edesc) tem que ser único;
- O nome dos equipamentos é obrigatório;
- Cada equipamento está associado a um departamento;
- Um equipamento pode não estar associado a nenhum tipo;
- O atributo "rating" só pode assumir os valores: A, B ou C

```
ou
CREATE TABLE department(
  departmentid NUMBER (2,0),
  dname VARCHAR2 (25) NOT NULL,
  PHONE VARCHAR2 (12),
  CONSTRAINT department_deptid_pk PRIMARY KEY (departmentid),
  CONSTRAINT department_dname_uk UNIQUE (dname)
);
```

2) Insira os registos nas tabelas.

DEPT

departmentid	dname	phone
4	Logistic	283123123
6	Production	289582471
7	Marketing	365159753

EQUIP

EquipID	Edesc	Purchasedate	Rating	DepartmentID	EtypeID
100	Computer	04/03/2000	A	7	25
121	Paint	02/12/2009	C	6	82
145	Printer	05/09/2020	A	7	25
146	Balance	5/9/2020	B	4	25

ETYPES

EtypeID	Etypename
8	Cleanning
25	Information Technology
82	Car Paint

3) Inserir 2 novos campos na tabela EQUIPMENT: selldate (não obrigatório) e price (do tipo 9999,99, tem que ser maior que 0, e não obrigatório).

Verificar estrutura:

COLUMN_NAME	DATA_TYPE	NULLABLE
EQUIPID	NUMBER(3,0)	No
EDESC	VARCHAR2(30 BYTE)	Yes
PURCHASEDATE	DATE	Yes
RATING	CHAR(1 BYTE)	Yes
DEPARTMENTID	NUMBER(2,0)	No
ETYPEID	NUMBER(2,0)	No
SELLDATE	DATE	Yes
PRICE	NUMBER(6,2)	Yes

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME
EQUIPMENT_DEPARTMENT_FK	Foreign_Key	(null)	SYSTEM	DEPARTMENT	DEPARTMENT_PK
EQUIPMENT_ETYPES_FK	Foreign_Key	(null)	SYSTEM	ETYPES	ETYPES_PK
EQUIPMENT_PK	Primary_Key	(null)	(null)	(null)	(null)
EQUIP_RATING_CK	Check	rating IN ('A', 'B', 'C')	(null)	(null)	(null)
SYS_C007993	Check	"EQUIPID" IS NOT NULL	(null)	(null)	(null)
SYS_C007994	Check	"DEPARTMENTID" IS NOT NULL	(null)	(null)	(null)
SYS_C007995	Check	"ETYPEID" IS NOT NULL	(null)	(null)	(null)
SYS_C008000	Check	price >0	(null)	(null)	(null)

4) Alterar o campo SELLDATE com data superior à PURCHASEDATE.

ou (sem nome restrição)

Verificar:

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
EQUIPMENT_DEPARTMENT_FK	Foreign_Key	(null)
EQUIPMENT_ETYPES_FK	Foreign_Key	(null)
EQUIPMENT_PK	Primary_Key	(null)
EQUIP_RATING_CK	Check	rating IN ('A', 'B', 'C')
SUPERIOR_DATA_COMPRA	Check	selldate>=purchasedate
SYS_C007993	Check	"EQUIPID" IS NOT NULL
SYS_C007994	Check	"DEPARTMENTID" IS NOT NULL
SYS_C007995	Check	"ETYPEID" IS NOT NULL
SYS_C008000	Check	price >0

5) Escreva uma consulta que mostre o nome dos equipamentos (Edesc) e o tipo (Etypename) de todos os equipamentos.

6) Escreva uma consulta que apresente o nome dos equipamentos (Edesc) e a data de compra dos equipamentos do departamento 7. A data deve apresentar o seguinte formato: dia-mês-ano.

edesc	purchase_date
Computer	03-04-2000
Printer	09-05-2020

7) Escreva uma consulta que identifique os equipamentos comprados depois de 2005 e que pertencem ao departamento 6.

8) Quais são os equipamentos sem avaliação (Rating)?

9) Escreva uma consulta que determine o número de meses entre a compra do equipamento 121 e a data atual.

todos os equipamentos:

10) Apagar todos os equipamentos da tabela EQUIPMENT que estão associados ao departamento 6.

11) Verificar se os dados foram corretamente eliminados. Utilize o comado ROLLBACK para anular a operação.

12) Construir uma script que permita obter o sresultado seguinte:

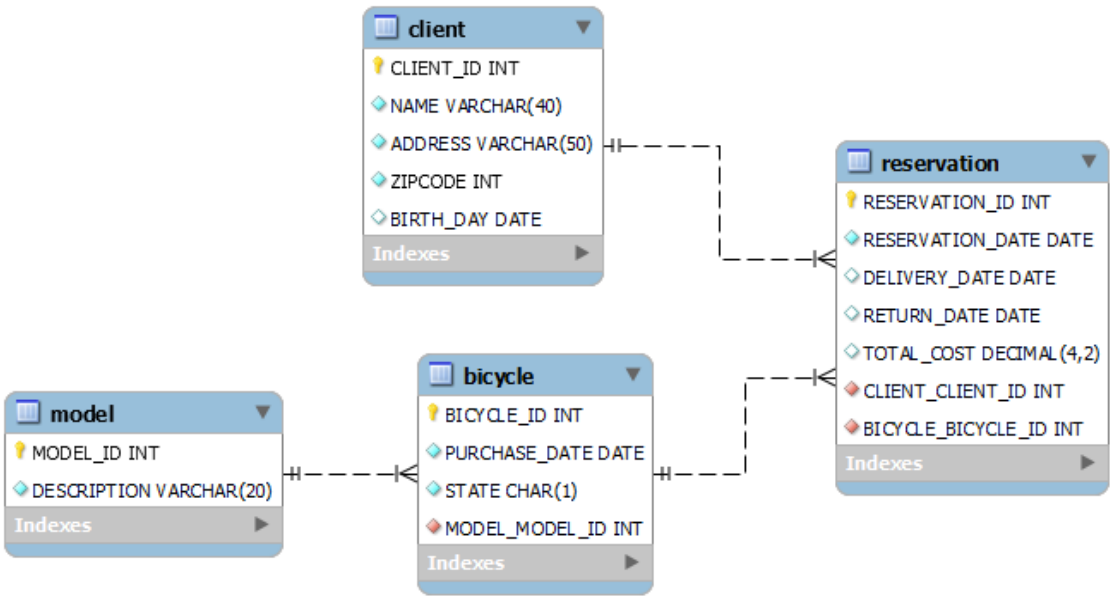
Equipamento	Tipo	Nome_Departamento	Data_Compra
Balance	Information Technology	Logistic	2020-09-05
Computer	Information Technology	Marketing	2000-03-04
Printer	Information Technology	Marketing	2020-09-05
Paint	Car painting	Production	2009-12-02

Exercício 6 | SQL / base de dados "Reserva bicicletas"

1) Crie a seguinte estrutura, tendo em atenção as PK, FK e restrições:

a) Criar schema `reserva_bicicletas`

b) Modelo relacional



c) Linhas

Tabela CLIENT

CLIENT_ID	NAME	ADDRESS	ZIPCODE	BIRTH_DAY
1	Joana Ferreira	Rua de cima 33	5300302	1990-05-03
2	Joao Pereira	Rua nova 44	5300845	NULL

Tabela MODEL

MODEL_ID	DESCRIPTION
1	BTT
2	ELECTRIC
3	FAMILIAR

Tabela BICYCLE

BICYCLE_ID	PURCHASE_DATE	STATE	MODEL_MODEL_ID
1	2019-01-01	0	2
2	2019-02-02	1	1
3	2019-03-03	1	2
4	2019-04-04	0	3
6	2022-02-05	0	2

Tabela RESERVATION

RESERVATION_ID	RESERVATION_DATE	DELIVERY_DATE	RETURN_DATE	TOTAL_COST	CLIENT_CLIENT_ID	BICYCLE_BICYCLE_ID
1	2020-01-01	2020-04-01	2020-04-01	90.50	2	3
2	2020-01-01	2020-05-01	2020-05-01	30.25	1	2
3	2021-06-11	2021-06-11	2021-06-13	60.75	2	2
4	2021-06-14	2021-06-14	2021-06-14	20.75	1	1
5	2021-06-14	NULL	NULL	0.00	1	1

d) Constraints / Restrições

Table Client

name	Data Type	Constraints
client_id	NUMBER (4)	PK
name	VARCHAR2(40)	Not null
address	VARCHAR2 (50)	Not null
zipcode	NUMBER (7)	Not null
birth_day	DATE	< data sistema

Table Reservation

name	Data Type	Constraints
reservation_id	NUMBER (5)	PK
reservation_date	DATE	Not null and data sistema
delivery_date	DATE	>= reservation_date
return_date	DATE	>= delivery_date
total_cost	NUMBER (4,2)	entre 0 e 100
client_client_id	NUMBER (4)	FK
bicycle_bicycle_id	NUMBER	FK

Table Bicycle

name	Data Type	Constraints
bicycle_id	NUMBER	PK
purchase_date	DATE	Not null
state	CHAR (1)	Not null
model_model_id	NUMBER (4)	FK

Table Model

name	Data Type	Constraints
model_id	NUMBER (4)	PK
description	VARCHAR2 (20)	

2) Criar as tabelas

3) Chaves estrangeiras/navegação/FK

4) Criar restrições em falta

Se pretender apagar o constraint

5) Inserir dados

a) Tabela CLIENT

b) Tabela MODEL

a) Tabela BICYCLE

a) Tabela RESERVATION

6) Escreva uma Script SQL que implemente, de forma adequada, a tabela RESERVATION, bem como as respetivas restrições de integridade.

7) Escreva uma script SQL que contabilize o total gasto por cliente que reservaram bicicletas do modelo "BTT" no ano 2021, ordenados por ordem decrescente do total. Output:

ID	NAME	Total
2	João Pereira	60.75

8) Escreva uma Script SQL que liste as bicicletas (bicycle_id, purchase_date, e model description) que nunca foram reservadas. Output:

bicycle_id	purchase_date	description
6	2022-02-05	ELECTRIC
4	2019-04-04	FAMILIAR

9) Escreva uma Script SQL que calcule a duração média arredondada das reservas (return_date - delivery_date) por modelo de bicicleta (exceto o modelo "Familiar"). Resultados ordenados por ordem crescente da média. Output:

Model Description	Media (Days)
Electric	1
BTT	2

10) Escreva uma Script SQL que permita saber quais são as reservas (e nome do cliente), que foram registadas a partir do ano 2021. Output:

reservation_id	Reservation Year	name
3	2021	Joao Pereira
4	2021	Joana Ferreira
5	2021	Joana Ferreira

ou

11) Escreva uma Script SQL que permita conhecer o nome dos clientes que efetuaram as 2 reservas mais caras do modelo BTT. Output:

reservation_id	client_client_id	name	total_cost	Model
3	2	Joao Pereira	60.75	BTT
2	1	Joana Ferreira	30.25	BTT