UNIVERSIDADE da MADEIRA

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Ano Letivo 2019/2020

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

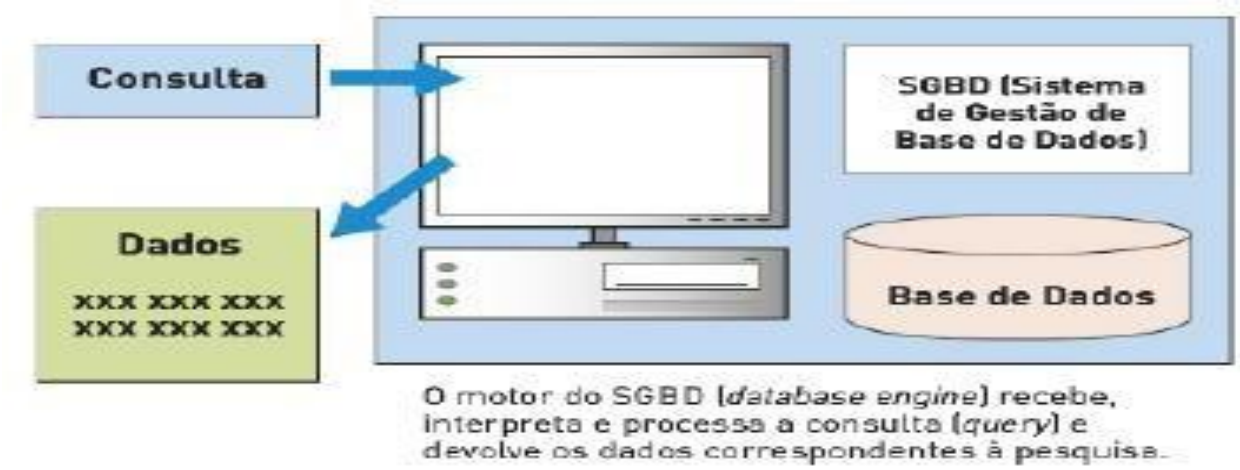
1 º Ano/2 º Semestre

Unidade Curricular: Sistemas Gestores de Bases de Dados I

Docente: Magno Andrade **Época:** Normal

SQL - MANIPULAÇÃO DE DADOS – PARTE 2

Para efectuar consultas a dados a bases de dados, utilizamos o comando SELECT.



COMANDO SELECT

O comando tem a seguinte **sintaxe**:

Estrutura básica de expressão de SQL:

SELECT (projeção da álgebra relacional)
FROM (lista das relações)
WHERE (predicado da seleção)

Ou seja:

SELECT $a_1, a_2, a_3, \ldots a_n$	(campos pretendidos)
FROM $r_1, r_2, r_3, \ldots r_n$	(tabelas dos campos pretendidos)
WHERE p	(condições a serem satisfeitas)

Sendo que a cláusula **WHERE** é opcional.

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
SELECT campo1, campo2, ... campon
FROM tabela 1 , tabela 2, .... tabela n
[WHERE Condição ]
[GROUP BY .....]
[HAVING ……..]
[ORDER BY …….];
```

Os parêntesis rectos indicam que essa componente é opcional.

✓ SELECCIONAR TODOS OS REGISTOS

Tendo em conta uma tabela **postal**, com os atributos **código** e **local**, pretende-se obter todos os registos desta tabela.

```
SELECT codigo, local
FROM postal;
```

Após a execução do comando, será obtido o seguinte resultado:

codigo	local
1000	LISBOA
1000	LISBOA
1500	LISBOA
2000	SANTARÉM
2040	RIO MAIOR
9000	FUNCHAL
4000	PORTO

✓ SELECCIONAR TODAS AS COLUNAS

Tendo em conta a tabela **pessoa**, com os atributos **id, nome, idade, salario, telefone** e **cod_postal**, pretendemos seleccionar todas as pessoas da tabela **pessoa**.

```
SELECT * FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	salario	telefone	cod_postal
42	António Dias	43	74000	789654	1000
5	Célia Morais	36	170000	123456	1100

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

32	Florinda Simões	35	147000		4000
37	Isabel Espada	28	86000		2040
49	José António	17	210000		1000
14	Nascimento Augusto	35	220000	456123	2300

As colunas são apresentadas pela ordem que são colocadas na cláusula SELECT, ou, se for colocado o asterisco, é a ordem das mesmas quando a tabela foi criada.

Pretendemos adicionar o **nome, salario** e **idade** das pessoas da tabela **pessoa**.

```
SELECT nome, salario, idade FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

nome	salario	idade
António Dias	74000	43
Célia Morais	170000	36
Florinda Simões	147000	35
Isabel Espada	86000	28
José António	210000	17
Nascimento Augusto	220000	35

✓ RESTRIÇÃO (CLÁUSULA WHERE)

Esta restrição permite restringir o número de linhas/registos a mostrar e tem de satisfazer a condição escolhida. Para as restrições, é necessário o uso de operadores que são apresentados abaixo:

Operadores Relacionais

Operador	Descrição	Exemplo	Resultado
=	Igual a	7=5	Falso
>	Maior que	7>5	Verdadeiro
<	Menor que	7<5	Falso
>=	Maior ou Igual que	7>=5	Verdadeiro
<=	Menor ou Igual que	7<=5	Falso

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

<> ou !=	Diferente	7<>5	Verdadeiro

Queremos obter, da tabela **pessoa**, todas as pessoas com **idade** igual a 35 anos.

```
SELECT *
FROM pessoa
WHERE idade = 35;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	salario	telefone	cod_postal
32	Florinda Simões	35	147000		4000
14	Nascimento Augusto	35	220000	456123	2300

Pretendemos obter o **id, nome** e **salario** das pessoas com **idade** maior ou igual a 18 anos.

```
SELECT id, nome, salario
FROM pessoa
WHERE idade >= 18;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	salario
42	António Dias	74000
5	Célia Morais	170000
32	Florinda Simões	147000
37	Isabel Espada	86000
14	Nascimento Augusto	220000

Operadores Lógicos

Operador	Exemplo
AND	condição ou atributo AND condição ou atributo
OR	condição ou atributo OR condição ou atributo
NOT	NOT condição

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Queremos seleccionar, da tabela **pessoa**, o **id, nome, idade** e **vencimento** de todas as pessoas com idade entre os 30 anos e os 40 anos.

```
SELECT id, nome, idade, salario as vencimento
FROM pessoa
WHERE idade >= 30 AND idade <= 40;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	vencimento
5	Célia Morais	36	170000
32	Florinda Simões	35	147000
14	Nascimento Augusto	35	220000

Queremos seleccionar, da tabela **pessoa**, o **id, nome** e **idade** das pessoas com idades maiores que os 40 anos e menores que os 30 anos.

```
SELECT id, nome, idade
FROM pessoa
WHERE idade < 30 OR idade > 40;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade
42	António Dias	43
37	Isabel Espada	28
49	José António	17

Queremos seleccionar, da tabela **pessoa**, o **id**, **nome** e **idade** das pessoas com idades maiores que os 40 anos e menores que os 30 anos.

```
SELECT id, nome, idade
FROM pessoa
WHERE NOT (idade >= 30 AND idade <= 40);
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade
42	António Dias	43

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

37	Isabel Espada	28
49	José António	17

Se neste caso, não for colocado os parêntesis, o resultado obtido é diferente.

```
SELECT id, nome, idade
FROM pessoa
WHERE NOT idade >= 30 AND idade <= 40;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade
37	Isabel Espada	28
49	José António	17

✓ OUTROS OPERADORES

BETWEEN -> permite especificar intervalos de valores.
IN -> permite especificar conjuntos de valores.
IS -> permite efectuar o tratamento de valores nulos (*NULL*) – *strings* vazias.
LIKE -> permite resolver alguns problemas naturais que existem quando se pretende comparar partes de *strings*.

✓ OPERADOR BETWEEN

Permite especificar intervalos de valores.

```
SELECT campo1, campo2, ... campoN
FROM tabela 1, tabela 2, .... tabela N
WHERE valor [NOT] BETWEEN valor1 AND valor2;
```

Queremos seleccionar, da tabela **pessoa**, o **id, nome, idade** e **vencimento** de todas as pessoas com idade entre os 30 e os 40 anos.

```
SELECT id, nome, idade, salario as vencimento
FROM pessoa
WHERE idade BETWEEN 30 AND 40;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	vencimento
5	Célia Morais	36	170000

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

32	Florinda Simões	35	147000
14	Nascimento Augusto	35	220000

Pretendemos obter, da tabela **pessoa**, o **id, nome, idade** e **vencimento** de todas as **pessoas** cuja idade não está compreendida entre os 30 e os 40 anos.

```
SELECT id, nome, idade, salario as vencimento
FROM pessoa
WHERE idade NOT BETWEEN 30 AND 40;
```

Ou

```
SELECT id, nome, idade, salario as vencimento
FROM pessoa
WHERE NOT idade BETWEEN 30 AND 40;
```

✓ Operador IN

Este operador permite especificar conjuntos de valores.

```
SELECT campo1, campo2, ... campoN
FROM tabela 1, tabela 2, .... tabela N
WHERE valor [NOT] IN (valor1, valor2, valorN);
```

Queremos obter, da tabela **postal**, os códigos postais existentes para os locais (LISBOA e FUNCHAL).

```
SELECT *
FROM postal
WHERE local IN ('LISBOA', 'FUNCHAL');
```

Após a execução do comando, será obtido o seguinte resultado:

codigo	**local**
1000	LISBOA
1000	LISBOA
1500	LISBOA
9000	FUNCHAL

Queremos obter, da tabela **postal**, os códigos postais existentes cujos locais não sejam (LISBOA e FUNCHAL).

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

```
SELECT *
FROM postal
WHERE local NOT IN ('LISBOA', 'FUNCHAL');
```

Após a execução do comando, será obtido o seguinte resultado:

codigo	local
2000	SANTARÉM
2040	RIO MAIOR
4000	PORTO

✓ Operador IS

Este operador permite efectuar o tratamento de valores nulos (*NULL*) – *strings* vazias. *NULL* é uma *string* vazia, não é zero.

A comparação com *NULL* tem de ser efectuada sempre com o operador IS, caso contrário vai devolver sempre falso.

```
SELECT campo1, campo2, ... campoN
FROM tabela 1, tabela 2, .... tabela N
WHERE valor IS [NOT] NULL;
```

Queremos seleccionar da tabela **pessoa**, os **nomes** das pessoas sem **telefone**.

```
SELECT nome
FROM pessoa
WHERE telefone IS NULL;
```

Após a execução do comando, será obtido o seguinte resultado:

nome
Florinda Simões
Isabel Espada
José António

Queremos obter, da tabela **pessoa**, os dados das pessoas com **telefone**.

```
SELECT id, nome, idade, salario as vencimento, cod_postal
FROM pessoa
WHERE telefone IS NOT NULL;
```

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	vencimento	cod_postal
42	António Dias	43	74000	1000
5	Célia Morais	36	170000	1100
14	Nascimento Augusto	35	220000	2300

✓ OPERADOR LIKE

Este operador permite resolver alguns problemas naturais que existem quando se pretende comparar *strings*.

A utilização do operador LIKE permite sempre fazer comparações de partes da *string*. Para isso acontecer utilizamos dois *WILDCARDS*.

WILDCARD	SIGNIFICADO
%	Qualquer string de zero ou mais caracteres
_ (underscore)	Um carácter qualquer

Consideramos uma tabela **mensagem**, com os atributos **msg_id** e **mensagem**, referentes a mensagens de uma empresa comercial.

Pretendemos obter os códigos de mensagem e descrição das mensagens que comecem pela letra T.

```
SELECT *
FROM mensagem
WHERE mensagem LIKE 'T%';
```

Após a execução do comando, será obtido o seguinte resultado:

msg_id	mensagem
80	Transportes
90	Telefonemas
105	Tratamentos

Queremos seleccionar os códigos de mensagem e descrição das mensagens que terminem em "as".

```
SELECT *
FROM mensagem
WHERE mensagem LIKE "%as";
```

Após a execução do comando, será obtido o seguinte resultado:

msg_id	mensagem
10	Comissão de Vendas

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

90	Telefonemas

Pretendemos obter os códigos de mensagem e descrição das mensagens que possuam a palavra "Vendas".

```
SELECT *
FROM mensagem
WHERE mensagem LIKE "%Vendas%";
```

Após a execução do comando, será obtido o seguinte resultado:

msg_id	mensagem
10	Comissão de Vendas
40	Vendas Extra

Pretendemos obter os códigos de mensagem e descrição das mensagens que cuja segunda letra seja "e".

```
SELECT *
FROM mensagem
WHERE mensagem LIKE "%_e%";
```

Após a execução do comando, será obtido o seguinte resultado:

msg_id	mensagem
40	Vendas Extra
90	Telefonemas

✓ PRECEDÊNCIA DOS OPERADORES

ORDEM	OPERADOR	SÍMBOLO
1ª	Parêntesis	( )
2ª	Multiplicação / Divisão	* /
3ª	Adição / Subtracção	+ -
4ª	NOT	
5ª	AND	
6ª	OR	

Queremos seleccionar, da tabela **pessoa**, o **id, nome** e **idade** das pessoas com idades maiores que os 29 anos e sem telefone e as pessoas com idades inferiores a 28 anos.

```
SELECT id, nome, idade
FROM pessoa
WHERE idade <= 27 OR
        idade >= 30 AND
        Telefone IS NULL;
```

É equivalente a:

```
SELECT id, nome, idade
FROM pessoa
WHERE idade <= 27 OR
        (idade >= 30 AND telefone IS NULL);
```

Queremos seleccionar, da tabela **pessoa**, o **id, nome** e **idade** das pessoas com idades maiores que os 29 anos e as pessoas com idades inferiores a 28 anos, que não possuam telefone.

```
SELECT id, nome, idade
FROM pessoa
WHERE (idade <= 27 OR idade >=30)
        AND telefone IS NULL;
```

✓ ORDENAÇÃO

A ordenação é realizada através da cláusula ORDER BY no comando SELECT. Quando esta cláusula aparece é sempre colocada no final do comando SELECT.

```
SELECT campo1, campo2, ... campoN
FROM tabela 1, tabela 2, .... tabela N
[WHERE Condição]
[GROUP BY .....]
[HAVING ……]
[ORDER BY Campo [ASC | DESC], Campo [ASC | DESC]…];
```

Onde Campo representa o nome de um Campo, uma Expressão ou a Posição pela qual se pretende ordenar o resultado da consulta.

ASC indica que a ordenação é **ASC**endente e **DESC** indica que a ordenação é **DESC**endente. Caso não seja indicado um tipo de ordenação, a mesma é efectuada por ordem ascendente.

✓ ORDENAR POR UMA COLUNA

Tendo em conta a tabela **pessoa**, queremos ordenar os dados por idade:

```
SELECT *
FROM pessoa
ORDER BY idade;
```

Ou

```
SELECT *
FROM pessoa
ORDER BY idade ASC;
```

Após a execução do comando, será obtido o seguinte resultado:

id	nome	idade	salario	telefone	cod_postal

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

49	José António	17	210000		1000
37	Isabel Espada	28	86000		2040
32	Florinda Simões	35	147000		4000
14	Nascimento Augusto	35	220000	456123	2300
5	Célia Morais	36	170000	123456	1100
42	António Dias	43	74000	789654	1000

✓ ORDENAR VÁRIAS COLUNAS

Considerando a tabela **comissao**, com os atributos **comissao_id, mensagem_id** e **valor**. Queremos ordenar os dados da tabela **comissao** por **comissao_id** e por **mensagem_id**.

```
SELECT *
FROM comissao
ORDER BY comissao_id, mensagem_id;
```

Após a execução do comando, será obtido o seguinte resultado:

comissao_id	mensagem_id	valor
14	10	10500
14	70	400
14	100	3750
25	10	2500
25	30	3370
37	40	14230
47	110	120

Queremos ordenar os dados da tabela **comissao**, em que **comissao_id** seja inferior a 30 de forma ascendente e o campo **valor** de forma descendente.

```
SELECT *
FROM comissao
WHERE comissao_id < 30
ORDER BY comissao_id ASC, valor DESC;
```

Após a execução do comando, será obtido o seguinte resultado:

comissao_id	mensagem_id	valor

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

14	10	10500
14	100	3750
14	70	400
25	30	3370
25	10	2500

✓ SELECÇÃO DE EXPRESSÕES

Tendo em conta a tabela **pessoa**. Pretende-se o **nome** e **idade** de todas as pessoas, seleccionando a **idade** que irão ter daqui a 3 anos, ordenando os dados de saída por **nome**.

```
SELECT nome, idade, idade + 3
FROM pessoa
ORDER BY nome;
```

Após a execução do comando, será obtido o seguinte resultado:

nome	idade	Expr1002
António Dias	43	46
Célia Morais	36	39
Florinda Simões	35	38
Isabel Espada	28	31
José António	17	20

Este cálculo é atribuído a uma Expr1002, que acaba por variar de sistema para sistema e é atribuída automaticamente.

Queremos o **nome** e **idade** de todas as **pessoas**, obtendo a **idade** que irão ter daqui a 3 anos, ordenando os dados pelo **nome** e renomear os campos **idade** para **idade_actual** e a **idade daqui a 3 anos** para **idade_em_2015.**

```
SELECT nome, idade AS 'idade_actual', idade + 3 AS 'idade_em_2015'
FROM pessoa
ORDER BY nome;
```

Após a execução do comando, será obtido o seguinte resultado:

nome	idade_actual	Idade_em_2015

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

António Dias	43	46
Célia Morais	36	39
Florinda Simões	35	38
Isabel Espada	28	31
José António	17	20

✓ ORDENAÇÃO E NULLS

A forma como o *NULL* é colocado no resultado ordenado de um SELECT depende de sistema para sistema. Alguns sistemas consideram o valor *NULL* menor que qualquer outro valor. Outros colocam o valor *NULL* sempre no topo dos valores, seja a ordenação ascendente ou descendente.

No MySQL, os valores *NULL* quando ordenados a ASC aparecem primeiro que quaisquer outros valores e no fim quando ordenados a DESC.

✓ ELIMINAÇÃO DE REPETIÇÕES (DISTINCT E ALL)

A cláusula DISTINCT colocada a seguir ao SELECT, tem como objectivo retirar dados duplicados. A cláusula ALL mostra todos os conjuntos de valores existentes, duplicados ou não.

Tendo em conta uma tabela **postal**, com os atributos **codigo** e **local**, pretendemos seleccionar todos os locais da tabela postal.

```
SELECT ALL local
FROM postal;
```

Ou

```
SELECT DISTINCT local
FROM postal;
```

Após a execução do comando, será obtido o seguinte resultado:

Cofinanciado por:

local
LISBOA
LISBOA
LISBOA
PORTO
FUNCHAL
TOMAR
PORTO

local
LISBOA
PORTO
FUNCHAL
TOMAR

Notas: a cláusula DISTINCT só pode ser colocada imediatamente a seguir ao SELECT.

✓ FUNÇÕES DE AGREGAÇÃO

São também designadas por Funções Estatísticas, têm por objectivo obter informações sobre o conjunto de linhas especificadas na cláusula WHERE ou sobre grupos de linhas indicadas na cláusula GROUP BY.

Função	Descrição
COUNT	Devolve o número de linhas.
MAX	Devolve o maior valor da coluna.
MIN	Devolve o menor valor da coluna.
SUM	Devolve a soma de todos os valores da coluna.
AVG	Devolve a média (AVeraGe) de todos os valores da coluna.

✓ FUNÇÃO COUNT

A função COUNT devolve o número total de linhas/registos que resultam de uma consulta SELECT. Pode ser utilizada de três formas distintas:

Função	Descrição
COUNT(*)	Devolve o número de linhas que resulta de um SELECT.
COUNT(coluna)	Devolve o número de ocorrências na coluna diferentes de NULL.

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

COUNT(DISTINCT coluna)	Devolve o número de ocorrências (sem duplicados) na coluna.

Tendo em conta a seguinte relação/tabela **pessoa**:

id	nome	idade	salario	telefone	cod_postal
42	António Dias	43	74000	789654	1000
5	Célia Morais	36	170000	123456	1100
32	Florinda Simões	35	147000		4000
37	Isabel Espada	28	86000		2040
49	José António	17	210000		1000
14	Nascimento Augusto	35	220000	456123	2300
75	Pedro Santos	42	235000		2040

Queremos saber quantas pessoas existem na tabela **pessoa**.

```
SELECT COUNT(*) AS Total
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Total
7

Queremos saber quantas pessoas existem na tabela **pessoa** e quantas pessoas possuem **telefone**.

```
SELECT COUNT(*) AS 'Total de pessoa',
        COUNT(Telefone) AS 'Total de telefones'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Total de pessoas	Total de telefones
7	3

Queremos saber quantos códigos postais diferentes existem na tabela **pessoa**.

```
SELECT COUNT(DISTINCT cod_postal) AS 'Cód. Postais diferentes'
```

```
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Cód. Postais diferentes
4

✓ Função MIN e MAX

A função MIN e MAX permite obter o menor e maior valor de uma determinada coluna.

Pretendemos saber qual o **salário** mais elevado e o **salário** mais baixo da tabela **pessoa**.

```
SELECT MAX(salario) AS 'Salário mais elevado',
       MIN(salario) AS 'Salário mais baixo'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Salário mais elevado	Salário mais baixo
235000	86000

Pretendemos saber a idade da **pessoa** mais nova e a idade da **pessoa** mais velha da tabela **pessoa**.

```
SELECT MAX(idade) AS 'Maior Idade',
       MIN(idade) AS 'Menor Idade'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Maior Idade	Menor Idade
43	17

Pretendemos saber o **nome** da primeira e da última **pessoa** da tabela **pessoa**, se a lista fosse ordenada alfabeticamente.

```
SELECT MAX(nome) AS 'Primeiro',
       MIN(nome) AS 'Último'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Primeiro	Último
António Dias	Pedro Santos

✓ FUNÇÃO SUM

A função SUM devolve a soma de uma determinada coluna.

Pretendemos saber o total de salários da tabela **pessoa**.

```
SELECT SUM(salario) AS 'Total de Salários'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Total de Salários
1162000

Pretendemos saber o total das idades da tabela **pessoa**.

```
SELECT SUM(idade) AS 'Total das idades'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Total de idades
236

Pretendemos saber o total das idades da tabela **pessoa**.

```
SELECT SUM(salario) AS 'Total de Salários',
       SUM(salario) * 1.03 AS 'Total de Salários com aumento'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Total de Salários	Total de Salários com aumento
1162000	1196860

✓ Função AVG

A função AVG devolve a média dos valores de uma determinada coluna.

Pretendemos saber a média das idades da tabela **pessoa**.

```
SELECT AVG(idade) AS 'Média de Idades'
FROM pessoa;
```

Após a execução do comando, será obtido o seguinte resultado:

Média de Idades
33.714285714285714

Existem funções que permitem formatar o resultado do SELECT, embora essas funções dependam de cada um dos sistemas de bases de dados.

Exemplo de duas funções:

Função	Descrição
ROUND	Arredonda por excesso um número para um número específico de casas decimais.
TRUNCATE	Arredondamento normal de um número para um número específico de casas decimais.

A sintaxe destas funções é a seguinte:

```
ROUND(número, número de casas decimais);
```

Exemplo:

```
SELECT ROUND(15.375, 1);
```

Este exemplo arredonda o número para cima e apenas a uma casa decimal.

Nota final: as funções MIN, MAX e COUNT podem ser utilizadas com qualquer tipo de dados. As funções SUM e AVG só podem ser aplicadas a campos numéricos. Se existirem valores *NULL* estes são ignorados por qualquer uma das funções.

✓ Funções De DATA

Em seguida são apresentadas algumas funções para manipulação de atributos do tipo de dados data.

Função	Descrição
YEAR	Retorna a parte do ano de uma determinada data (entre 1000 a 9999).

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

MONTH	Retorna a parte do mês de uma determinada data (entre 1 a 12).
DAY	Retorna o dia do mês de uma determinada data (entre 1 a 31).
CURTIME	Retorna o tempo actual em formato “HH-MM-SS" (*string*) ou em HHMMSS.uuuuuu (numérico).
HOUR	Retorna a parte da hora de uma determinada data/tempo.
MINUTE	Retorna a parte do minuto de uma determinada data/tempo.
SECOND	Retorna a parte do segundo de uma determinada data/tempo.

A sintaxe destas funções é a seguinte:

SELECT YEAR(data_de_nascimento);

Para as restantes funções a sintaxe é semelhante.

✓ Funções De STRING

Em seguida são apresentadas algumas funções para manipulação de atributos do tipo de dados *string*.

Função	**Descrição**
UCASE	Converte uma *string* para letras maiúsculas.
LCASE	Converte uma *string* para letras minúsculas.
CONCAT	Junta uma ou mais *strings*.
REVERSE	Retorna o inverso da *string*.

A sintaxe destas funções é a seguinte:

SELECT UCASE(“magno”)

Para as restantes funções a sintaxe é semelhante.

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu