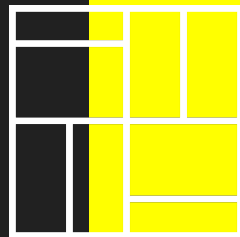




# DATABASES FUNDAMENTALS

TUTORA

Natacha Cabral





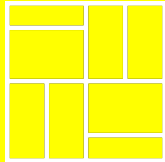
# 1. SQL Basics



# Criar todas as tabelas da bd ginásio

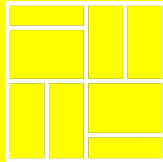
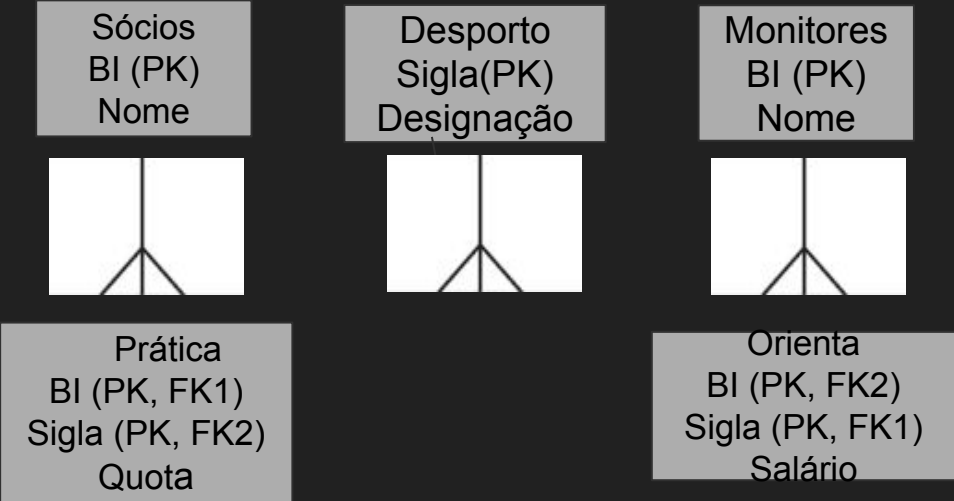
• Gym example:

Socios		Monitores		Desportos		Pratica			Orienta		
BI	Nome	BI	Nome	Sigla	Designação	BI	Sigla	Quota	BI	Sigla	Salario
6078	Ana	9876	Luís	KB	Kickbox	6078	AE	25	1234	KB	40
5819	Rui	1234	Joana	NT	Natação	5819	KB	30	1234	NT	30
4526	Nuno			AE	Aeróbica	4526	KB	30	9876	NT	30
3955	Rita					4526	NT	20	9876	AE	35
9999	José					3955	KB	30			
						3955	NT	20			
						3955	AE	25			
						9876	KB	0			



# Criar todas as tabelas da bd ginásio

Exemplo de um Diagrama Lógico  
Ginásio



## Criar tabelas com SQL

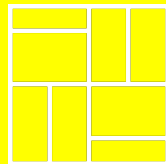
```
CREATE TABLE public.socios (  
    bi character varying(20) PRIMARY  
    KEY,  
    nome character varying(100) NOT  
    NULL  
);
```

```
CREATE TABLE public.pratica (  
    bi character varying(20)  
    REFERENCES public.socios(bi),  
    sigla character varying(10)  
    REFERENCES  
    public.desportos(sigla),  
    quota integer NOT NULL  
);
```

```
CREATE TABLE public.monitores (  
    bi character varying(20)  
    PRIMARY KEY,  
    nome character varying(100)  
    NOT NULL  
);
```

```
CREATE TABLE public.orienta (  
    bi character varying(20) REFERENCES  
    public.monitores(bi),  
    sigla character varying(10) REFERENCES  
    public.desportos(sigla),  
    salario integer NOT NULL  
);
```

```
CREATE TABLE public.desportos (  
    sigla character varying(10) PRIMARY  
    KEY,  
    designacao character varying(100) NOT  
    NULL  
);
```



# Criar tabelas com SQL

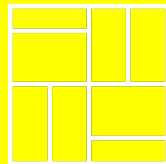
```
CREATE TABLE public.orienta (  
    bi character varying(20) REFERENCES  
public.monitores(bi),  
    sigla character varying(10) REFERENCES  
public.desportos(sigla),  
    salario integer NOT NULL  
);
```

```
INSERT INTO public.orienta (bi, sigla, salario)  
VALUES  
    ('1234', 'KB', '40'),  
    ('1234', 'NT', '30'),  
    ('9876', 'NT', '30'),  
    ('9876', 'AE', '35');
```

A palavra-chave REFERENCES é usada no SQL, especialmente no PostgreSQL, para definir uma **chave estrangeira** (foreign key). Ela estabelece uma **relação** entre uma coluna (ou conjunto de colunas) e uma tabela com uma coluna correspondente em outra tabela, garantindo que os valores da primeira tabela existam na segunda.

## Significado e Uso de REFERENCES

Quando usamos REFERENCES ao definir uma coluna, estamos a dizer que essa coluna **deve corresponder a um valor existente** na coluna referenciada de outra tabela. Isso ajuda a manter a integridade referencial no banco de dados, evitando, por exemplo, que uma tabela contenha valores de identificação que não tenham correspondência em outra tabela.



# Criar manualmente

Resultado final  
ver na tabela, lado direito do rato,  
properties

Create - Table

General Columns Advanced Constraints Partitions Parameters Security SQL

Name	Columns	Referenced Table
id	(id) -> (id)	public.socios

General Definition Columns Action

Columns

Local column

Select an item...

References

public.socios

Referencing

Select an item...

Add

Local	Referenced	Referenced Table
id	id	public.socios

Close Reset Save

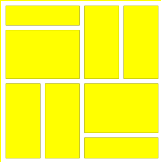
pratica

General Columns Advanced Constraints Parameters Security SQL

Primary Key Foreign Key Check Unique Exclude

Name	Columns	Referenced Table
pratica_bi_fkey	(bi) -> (bi)	public.socios
pratica_sigla_fkey	(sigla) -> (sigla)	public.desporto

Não esquecer selecionar add depois  
de colocar os campos e save



# Inserir valores nas com SQL

```
INSERT INTO public."socios"("BI", "Nome")  
VALUES  
  ('6078', 'Ana'),  
  ('5819', 'Rui'),  
  ('4526', 'Nuno'),  
  ('3955', 'Rita'),  
  ('9999', 'José');
```

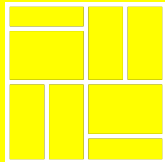
```
INSERT INTO public.monitores (bi, nome)  
VALUES  
  ('9876', 'Luís'),  
  ('1234', 'Joana')  
  
select * from public.monitores
```

```
INSERT INTO public.desportos (sigla, designacao)  
VALUES  
  ('KB', 'Kickbox'),  
  ('NT', 'Natação'),  
  ('AE', 'Aerobica');
```

```
INSERT INTO public.pratica (bi, sigla, quota)  
VALUES  
  ('6078', 'AE', '25'),  
  ('5819', 'KB', '30'),  
  ('4526', 'KB', '30'),  
  ('4526', 'NT', '20'),  
  ('3955', 'KB', '30'),  
  ('3955', 'NT', '20'),  
  ('3955', 'AE', '25');  
-- ('9876', 'KB', '0');
```

```
INSERT INTO public.orienta (bi, sigla, salario)  
VALUES  
  ('1234', 'KB', '40'),  
  ('1234', 'NT', '30'),  
  ('9876', 'NT', '30'),  
  ('9876', 'AE', '35');
```

A última tabela dá erro porque o valor 9876 não existe na tabela socio



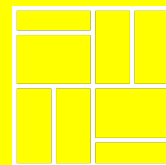


## Conceitos Básicos

- **SELECT... será o início da tua jornada SQL!**

**Select é utilizado para mostrar a informação que se procura**

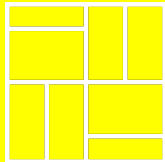
**Regra 1 - começar por visualizar as tabelas  
(não esquecendo o limit)**



# Select

```
Select col1, col2,...  
From table1
```

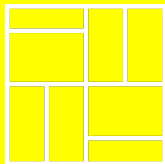
1. Apresenta toda a informação da tabela sócios



- `select * from socios;`

Result

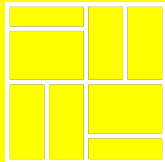
BI	Nome
6078	Ana
5819	Rui
4526	Nuno
3955	Rita
9999	José



## Conceitos Básicos

À instrução de SELECT seguem se várias cláusulas, apresentadas à direita:

- WHERE
- DISTINCT
- ORDER BY
- GROUP BY
- LIMIT
- FETCH
- HAVING
- JOIN (INNER JOIN / LEFT JOIN / FULL OUTER JOIN / CROSS JOIN)
- UNION
- INTERSECT
- EXCEPT



## Conceitos Básicos

**Where:** Utilizado para filtrar linhas (restrição)

**Like:** Utilizado para procurar strings

**Distinct :** Utilizado para remover duplicados (eliminação de repetições)

**In:** Utilizado para procurar valores pertencentes a um conjunto

**Order by:** Utilizado para organizar o output (ordenação)

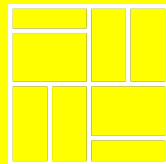
**Group by:** Utilizado para agregação de linhas e cálculo de valores

**Having:** Utilizado para selecionar linhas de agregação

**Wildcards = %** - Usar para tudo o que está a frente ou atrás de

**Limit:** Quando só queremos um determinado número de dados

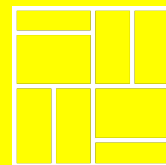
**Offset:** quando queremos retornar um determinado número de registros mas não queremos o primeiro (por exemplo LIMIT 3 OFFSET 1)



## Filtrar linhas

```
Select col1, col2  
From table1  
Where <condition>
```

2. Lista de todos os BI e nome dos membros, onde os número do Bi é menor do que 6000

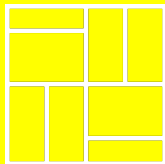


## Filtrar linhas

```
select bi, nome from socios where bi < '6000';
```

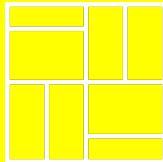
Results

BI	Nome
5819	Rui
4526	Nuno
3955	Rita



# Operadores relacionais

=	Testa a igualdade
!=	Testa a diferença
<	Testa se é menor
>	Testa se é maior
<=	Testa se é menor ou igual
>=	Testa se é maior ou igual
BETWEEN	Testa se o valor está compreendido num intervalo
IN	Testa se o valor de uma linha está contido num conjunto de valores especificados
EXISTS	Testa se algum registo existe dadas as condições fornecidas
LIKE	Testa se um valor corresponde à <i>string</i> especificada
IS NULL	Testa para valores NULL
IS NOT NULL	Testa para todos os valores que não NULL





## Procurar por strings

3.1 Lista de todos os BI e nome dos membros, onde os nomes começam por 'R'

3.2 Lista de todos os BI e nome dos membros, onde os nomes acabam com 'a'

Compare strings with **like** :

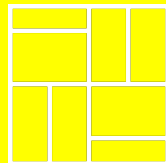
% look for any string with 0 or more characters :

name **like** 'M%' (Oracle, MySQL)

'Marina' will be na output

Compare strings with **=**, looks for the exact match :

name **=** 'M%' is true if the name is 'M%'



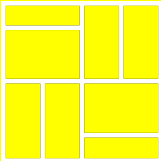
# Procurar por strings

select \* from public.socios  
where nome like 'R%'

Data Output Messages Notifications		
<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>		
	bi [PK] character varying (20)	nome character varying (100)
1	5819	Rui
2	3955	Rita

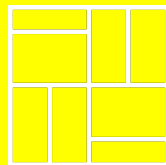
select \* from public.socios  
where nome like '%a'

Data Output Messages Notifications		
<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>		
	bi [PK] character varying (20)	nome character varying (100)
1	6078	Ana
2	3955	Rita



## Procurar por strings e por operadores relacionais

3.3 Lista de todos os BI e nome dos membros, onde os nomes começam por 'R' ou o número BI está entre 4000 e 5000.

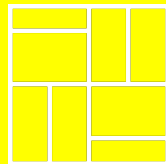


## Procurar por strings e por operadores relacionais

```
select bi, nome  
from socios where nome like 'R%' or bi between '4000' and '5000';
```

Results

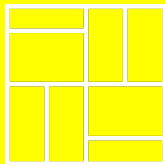
BI	Nome
5819	Rui
4526	Nuno
3955	Rita



## Remover duplicados

```
Select distinct col1, col2  
From table1
```

4. Selecciona as siglas dos desportos que possuem participantes



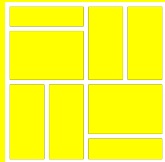
## Remover duplicados

select sigla from  
pratica;

Results	
Sigla	
AE	
KB	
KB	
NT	
KB	
NT	
AE	
KB	

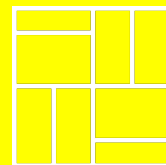
select distinct sigla from  
prática;

Results	
Sigla	
AE	
KB	
NT	



## Expressões Aritméticas

5. Quais seriam os salários acima dos 40€ se aumentarmos 20%?

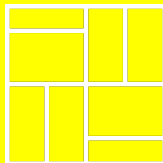


## Expressões Aritméticas

```
select bi, sigla, salario*1.2 as  
"Novo salário" from public.orienta  
where salario*1.2 > '40'
```

Results

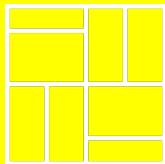
BI	Sigla	Novo Salario
1234	KB	48
9876	AE	42





## Valores pertencentes a um conjunto - IN

6. lista os Bi dos membros e as siglas de quem pratica aeróbica ou natação



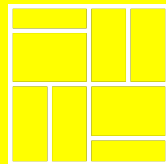
## Valores pertencentes a um conjunto - IN

```
select bi, sigla  
from public.pratica  
where sigla IN ('AE','NT');
```

```
select bi, sigla  
from public.pratica  
where sigla = 'AE' or sigla = 'NT';
```

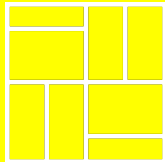
Results

BI	Sigla
3955	AE
3955	NT
4526	NT
6078	AE



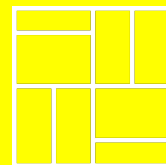
## Ordenar um output - order by

```
Select col1, col2  
From table1  
Where <condition>  
Order by col_1;
```



## Ordenar um output - order by

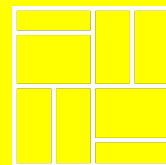
7. Obtenha uma lista de siglas esportivas e BI dos membros, classificados em ordem crescente por sigla e decrescente por BI



## Ordenar um output - order by

```
select sigla, bi  
from public.pratica  
order by sigla , bi desc
```

Results	
Sigla	BI
AE	6078
AE	3955
KB	9876
KB	5819
KB	4526
KB	3955
NT	4526
NT	3955



## Funções de agregação

Permitem obter informação sobre um conjunto de registros;

**Count:** Utilizado para contar linhas

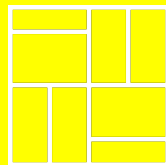
**Count Distinct:** Utilizado para contar linhas sem duplicados

**avg:** Utilizado para fazer a média

**sum:** Utilizado para fazer somatórios

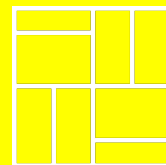
**max:** Utilizado para procurar valores máximos

**min:** Utilizado para procurar valores mínimos



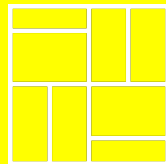
## Funções de agregação

```
Select col1, col2,..., count(colx)  
From table1  
Where <condition>  
Group by col1, col2, ...;
```



## Funções de agregação

8. Obtenha o salário médio, o número de salários e o total de salários, bem como o maior e o menor salário





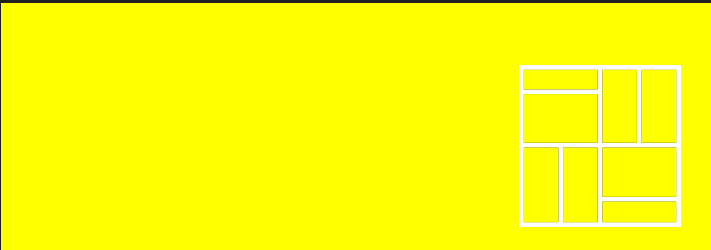
# Funções de agregação

select  
  
avg(salario) as mean,  
count(\*) as number,  
sum(salario) as total,  
max(salario) as  
max,min(salario) as min  
  
from orienta

SELECT  
    ROUND(AVG(salario), 2)  
AS mean,  
    COUNT(\*) AS number,  
    SUM(salario) AS total,  
    MAX(salario) AS max,  
    MIN(salario) AS min  
FROM orienta;

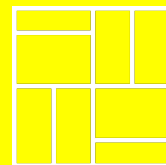
Results

Mean	Number	Total	Max	Min
33,75	4	135	40	30



## Funções de agregação

9. Calcule o valor total pago por cada associado (bi e valor)

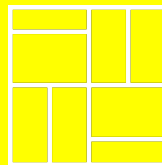


## Funções de agregação

```
select bi, sum(quota) from pratica  
group by bi  
  
order by sum(quota) desc
```

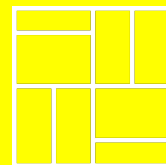
Results	
BI	Sum(quota)
3955	75
4526	50
5819	30
6078	25
9876	0

*É essencial ter no group by todas as colunas presentes na instrução select*



## Funções de agregação

10. Calcule o valor total pago por cada membro, mas apenas para casos acima de 40€



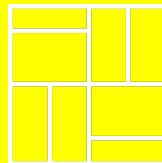
## Funções de agregação

~~select bi, sum(quota)  
from pratica  
where sum(quota) > 40  
group by bi~~

select bi, sum(quota)  
from pratica  
group by bi  
having sum(quota) > 40

*O having seleciona linhas  
da agregação como o  
where seleciona linhas  
da tabela base*

Data Output		Messages	Notifications
	bi character varying (20)	sum bigint	
1	3955	75	
2	4526	50	



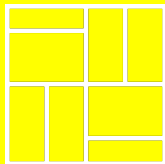
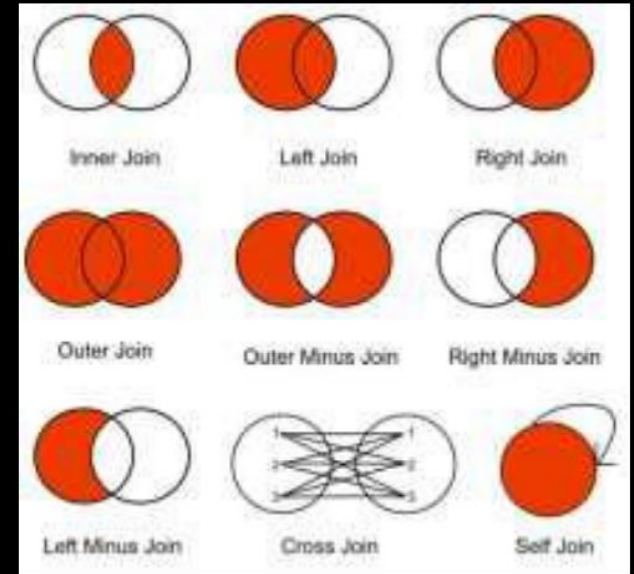
# JOINS

JOINS mais comuns entre tabelas/views:

**LEFT JOIN** – devolve todos os registos da tabela da esquerda, e os registos correspondentes da tabela da direita

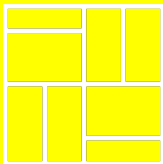
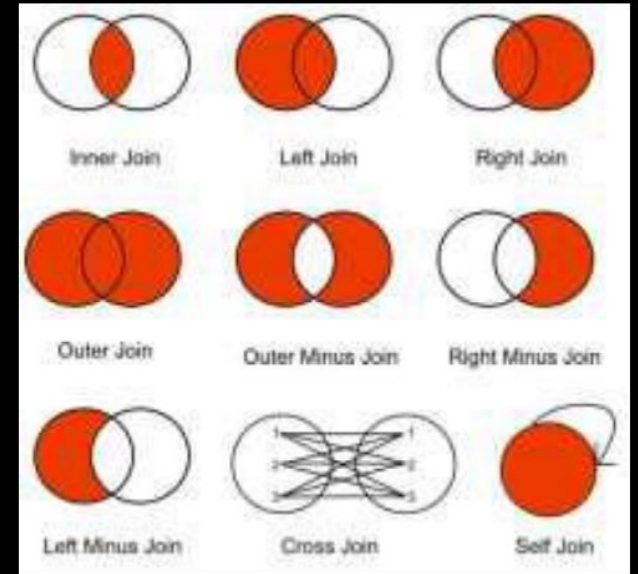
**INNER JOIN** – apenas devolve registos que estão presentes nas duas tabelas/views

**RIGHT JOIN** – devolve todos os registos da tabela da direita, e os registos correspondentes da tabela da esquerda



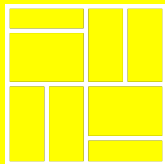
# JOINS

```
Select *  
From table1 t1  
(left/right) Join table2 t2 on t1.col1 = t2.col3
```



## JOINS

11. Liste o nome dos sócios e a sigla do desporto dos membros que praticam aeróbica e natação





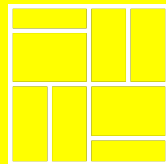
## JOINS

11. Liste o nome dos sócios e a sigla do desporto dos membros que praticam aeróbica e natação

```
select nome, sigla  
from socios  
join pratica on  
socios.bi=pratica.bi where  
sigla in ('AE', 'NT')
```

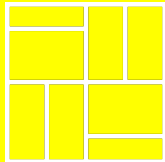
Results

Nome	Sigla
Ana	AE
Nuno	NT
Rita	NT
Rita	AE



## JOINS

12. Liste o nome dos sócios e a designação do desporto dos membros que praticam aeróbica e natação



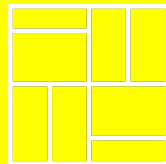
## JOINS

12. Liste o nome dos sócios e a designação do desporto dos membros que praticam aeróbica e natação

```
select s.nome, d.designacao
from socios s
join pratica p on s.bi=p.bi
join desportos d on p.sigla = d.sigla
where p.sigla in ('AE', 'NT')
order by 1 asc
```

Results

Nome	Designação
Ana	Aeróbica
Nuno	Natação
Rita	Natação
Rita	Aeróbica



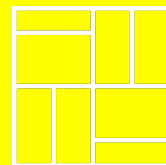
## União e intersecção de tabelas

### UNION Statement

```
Select col1, col2  
From table1  
Union (all)  
Select col3, col4  
From table2;
```

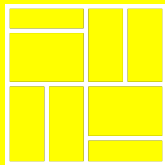
### INTERSECT Statement

```
Select col1, col2  
From table1  
Intersect  
Select col3, col4  
From table2;
```



## União e intersecção de tabelas

13. Lista todos os nomes dos membros e monitores













## União e intersecção de tabelas

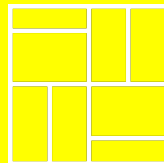
### 13. Lista todos os nomes dos membros e monitores

(select nome from socios)

Union

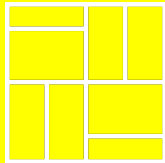
(select nome From  
monitores)

Data Output		Messages	Notifications
        			
	<b>nome</b> character varying (100) 		
1	Rita		
2	Rui		
3	Ana		
4	Joana		
5	Nuno		
6	José		
7	Luís		



## União e intersecção de tabelas

13. Liste os nomes comuns dos membros e monitores?



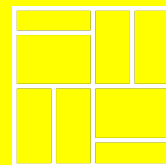
## União e intersecção de tabelas

13. Liste os nomes comuns dos membros e monitores?

(select nome from socios)

intersect

(select nome From  
monitores)







## 2. SQL em prática - Individual



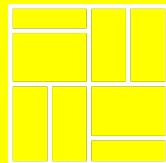
## Exercício A - Select

- Exemplo: `SELECT * FROM actor;`
  1. Select only the first name.
  2. Select only first and last names.
  3. Select only distinct first name.
  4. Select only distinct first names that are different from Nick.

### Dica:

```
SELECT col1, col2  
FROM table1  
WHERE <condition>;
```

```
/*  
comment  
*/
```



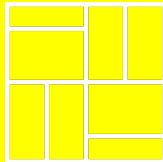
## Exercício B - ORDER BY

1. Select all actor's data sorted by last\_name.
2. Select all actor's data sorted by last name but by descending order.
3. Select first and last names, sorted by last name by ascending order, from actors whose first name starts with a 'B'.

Dica:

```
SELECT col1, col2  
FROM table1  
WHERE <condition>  
ORDER BY col2 ASC;
```

```
/*  
comment  
*/
```



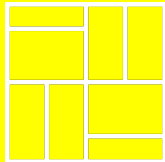
## Exercício C - GROUP BY

1. Number of records of the actor's table.
2. How many actors have the same first name, sorted by first name but descending.
3. Create an alias for the count column named 'howmany'.
4. Reuse the previous query and sort it by 'howmany' in descending order and by 'first\_name' ascending.

### Dica:

```
SELECT col1, col2, ..., count(colx) c
FROM table1
WHERE <condition>
GROUP BY col1, col2, ...;

/*
comment
*/
```

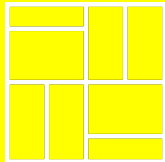


## Exercício D - LIMIT, OFFSET & FETCH

1. Select the first 6 rows of actor's table.
2. Select 4 rows (from the 3rd until the 6th) from actor's table.
3. Same as before but using FETCH.

Dica:

```
SELECT col1, col2, ...  
FROM table1  
WHERE <condition>  
LIMIT n;  
-- OR (this is also a comment)  
SELECT col1, col2, ...  
FROM table1  
WHERE <condition>  
OFFSET n row  
FETCH first n row only;
```

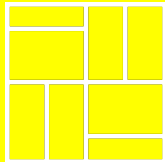


## Exercício E - HAVING

1. For each 'customer\_id' in payment's table, show the total of the amounts paid.
2. Same as before, filtering the customers that spent more than 200€

### Dica:

```
SELECT col1, col2, ..., COUNT(colx)
FROM table1
WHERE <condition>
GROUP BY col1, col2, ...
HAVING COUNT(colx) > y;
```

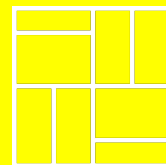


## Exercício F - UNION & UNION ALL

1. Count the number of customers.
2. Count the number of actors.
3. List the distinct first names of actors and customers in the same column.
4. List the first names of actors and customers in the same column.

Dica:

```
SELECT col1, col2  
FROM table1  
UNION (ALL)  
SELECT col3, col4  
FROM table2;
```

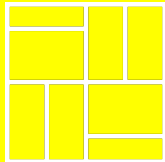


## Exercício G – INTERSECT & EXCEPT

1. List the common first names of both actors and customers in the same column.
2. List the first names of actors that don't exist on customer table.

Dica:

```
SELECT col1, col2  
FROM table1  
EXCEPT  
SELECT col3, col4  
FROM table2;
```



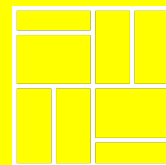


## Exercício H – BETWEEN, IN, IS NULL

1. Select customer information where the customer\_id is bigger than 9 and lower than 15.
2. Select customer information where the first name is Lisa or Marion.
3. Select staff information where picture is NULL.
4. Select staff information where picture is not NULL.

Dica:

```
SELECT col1, col2  
FROM table1  
WHERE col1 BETWEEN x AND y;
```

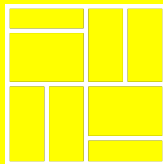
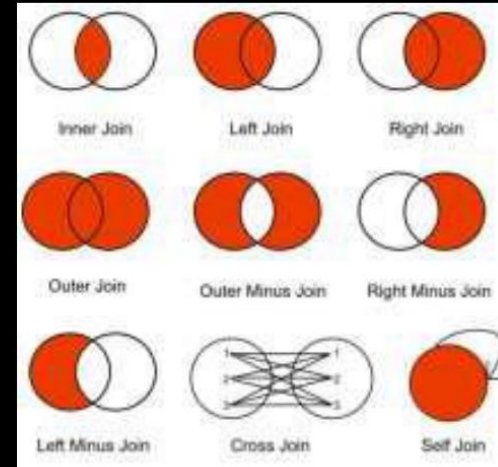


Para relembrar...

## JOINS

JOINS mais comuns entre tabelas/*views*:

- LEFT JOIN – devolve todos os registos da tabela da esquerda, e os registos correspondents da tabela da direita
- INNER JOIN – apenas devolve registos que estão presents nas duas tabelas/*views*
- RIGHT JOIN – devolve todos os registos da tabela da direita, e os registos correspondentes da tabela da esquerda

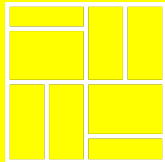


## Exercício I – JOIN

1. For each customer in customer table, show first and last names and the corresponding address, from address table.
2. Identify the existing 'address\_id's on table address which have no corresponding customer. (without using JOIN)
3. List 'address\_id', 'address', 'first\_name' for customers with 'address\_id' <= 7, sorted by 'address\_id'.

### Dica:

```
SELECT *  
FROM table1 t1  
JOIN table2 t2 ON t1.col1 = t2.col2;
```



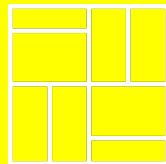
## Exercício I – JOIN

4. List 'address\_id', 'address', 'first\_name' for customers with 'address\_id' <= 7, sorted by 'address\_id', using LEFT JOIN.

Also, try to adapt the same query now using RIGHT JOIN.

Dica:

```
SELECT *  
FROM table1 t1  
JOIN table2 t2 ON t1.col1 = t2.col2;
```



## Exercício J – CASE

1. For each record having 'address2' as NULL, on address table, present:

- address2
- address3, as result of using COALESCE to replace NULL values by 'Unknown' on address2
- address
- last char, as result of (Using case) is 'e', return 'E';
- if address' last character is 'd', return 'D';
- if address' last character is anything else, return 'X'

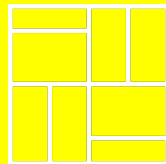
**coalesce:** permite fornecer um valor padrão ou substituto quando são encontrados nulls e ajuda a garantir que dados significativos sejam exibidos nos resultados de pesquisa mesma na presença de valores NULL

**case when:**

A instrução CASE é a maneira do SQL lidar com a lógica se/então se algo ... então X

Dica:

```
SELECT CASE WHEN cond1 THEN res1
        WHEN ...
        ELSE resx
        END
FROM table1 t1;
```



## Exercícios – K

1. For each customer first\_name in lower case, list the year of his last\_update.
2. For each customer first\_name, list: store\_id, active, store\_id+active (eg.: "1+2"), and store\_idactive (eg.: "12").

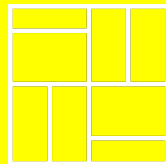
**Cast:** Podemos converter dados de um tipo para outro dentro de uma instrução SQL. Um exemplo de uso dessa função, por exemplo, seria a conversão de uma coluna do tipo numérico para VarChar para que a mesma possa ser concatenada com outra coluna ou valor durante o resultado de uma instrução Select.

Dica:

`LOWER(col1)`

`EXTRACT(part FROM col1)`

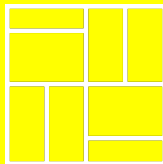
`CAST(expression AS datatype)`



## Exercícios – K

3. For each customer last\_name, list the film titles he rented and the corresponding language name and rental rate.

Records should be sorted starting on the most expensive ones. Only show the films with a rental rate equal or bigger than 2.99€.





# DATABASES FUNDAMENTALS