iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 2: Conceitos e Estrutura de Bases de Dados

Aula 9

SQL: Funções de Agregação, Group by e Having; Subqueries

UPskill

O que aprendemos ontem?

- O que é o SQL.
- Como podemos fazer consultas, atualizações e eliminações numa base de dados:
  - SELECT
  - UPDATE
  - DELETE
- Aplicar filtros em querys criadas:
  - WHERE
  - ORDER BY

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Hoje vamos continuar a aprender mais sobre SQL!

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Funções de Agregação

Quando utilizadas no SELECT produzem como output **um único valor**. Uma função de agregação aplica-se a um conjunto de registos (linhas) e produz um valor.

- **COUNT(coluna ou *)** - Devolve o número de linhas/registos/tuplos
- **MAX(coluna)** - Devolve o maior valor da coluna
- **MIN(coluna)** - Devolve o menor valor da coluna
- **SUM(coluna)** - Devolve a soma de todos os valores da coluna
- **AVG(coluna)** - Devolve a média (AVeraGe) de todos os valores da coluna

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Funções de Agregação

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

SELECT COUNT(numero) FROM estudante
- 7

SELECT COUNT (*) FROM estudante
- 7

SELECT COUNT(curso) FROM estudante
- 6

SELECT AVG(numero) FROM estudante
- 20004

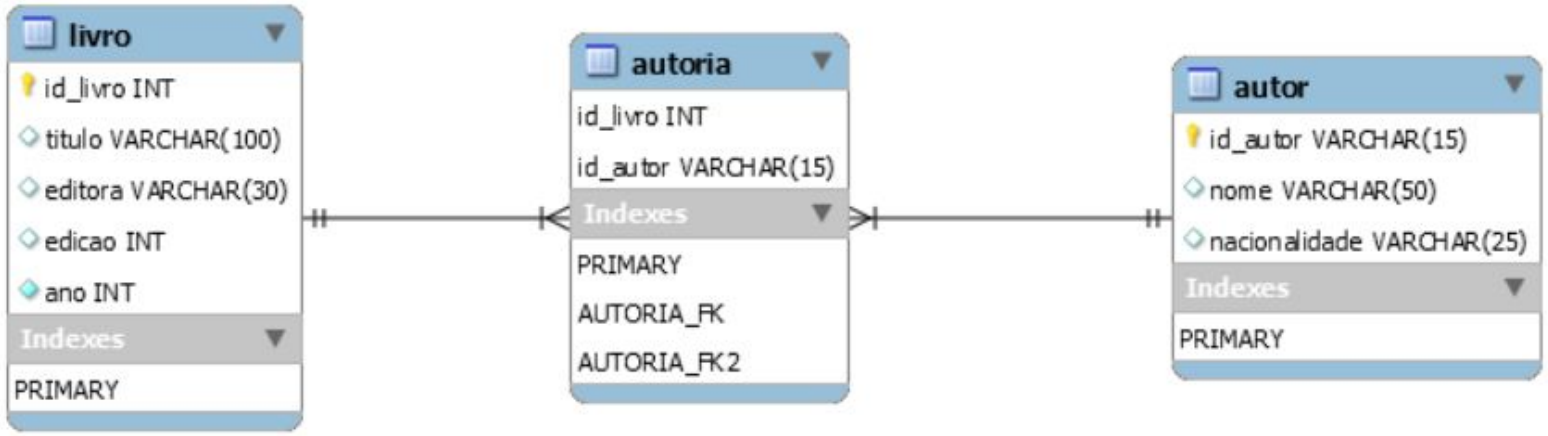
SELECT MIN(numero) FROM estudante
- 20001

SELECT MAX(numero) FROM estudante
- 20007

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Ficheiros a usar para os exercícios

- Livros - Estrutura v2.sql
- Livros - Dados v2.sql



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Exercício 1

Na base de dados dos livros, descubra:

- Quantos livros existem?
- Em que ano foi lançado o livro mais antigo?
- Em que ano foi lançado o livro mais recente?
- Quantos livros foram lançados em 2003?
- Quantos livros foram lançados no século XX?

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Exercício 1 - Solução

Na base de dados dos livros, descubra:

- Quantos livros existem?
  - SELECT COUNT(*) AS total FROM livro
- Em que ano foi lançado o livro mais antigo?
  - SELECT MIN(ano) FROM livro
- Em que ano foi lançado o livro mais recente?
  - SELECT MAX(ano) FROM livro
- Quantos livros foram lançados em 2003?
  - SELECT COUNT(*) FROM livro WHERE ano=2003
- Quantos livros foram lançados no século XX?
  - SELECT COUNT(*) FROM livro
    WHERE ano>1900 AND ano<=2000

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

GROUP BY

O **GROUP BY** permite agrupar linhas com os mesmos valores numa coluna.

É normalmente usado com funções de agregação.

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

```
SELECT COUNT(numero) AS total_alunos, curso
FROM estudante
GROUP BY curso
```

Total_Alunos	Curso
1	Economia
2	Informática
1	Marketing
1	Sociologia
1	História
1	Null

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

HAVING

O **HAVING** é a cláusula adicionada ao SQL porque o WHERE não funciona em funções de agregação.

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

```
SELECT COUNT(numero) AS total_alunos, curso
FROM estudante
GROUP BY curso
HAVING total_alunos > 1
```

Total_Alunos	Curso
2	Informática

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

WHERE vs HAVING

WHERE	HAVING
A cláusula WHERE tem como finalidade seleccionar registros.	A cláusula HAVING tem como finalidade seleccionar grupos que obedeçam a determinadas características de grupo.
A cláusula WHERE nunca contém funções de agregação.	A cláusula HAVING deve sempre conter funções de agregação.
A cláusula WHERE tem prioridade relativamente à cláusula HAVING.	

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Exercício 2

Na base de dados dos livros:

- Liste o número de livros publicados por ano por ordem descendente.
- Liste os anos que tiveram mais de 3 publicações de livros.
- Liste os anos em que a editora Dom Quixote publicou apenas um livro.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Exercício 2

Na base de dados dos livros:

- Liste o número de livros publicados por ano por ordem descendente.
  - SELECT ano, COUNT(*) AS total FROM livro GROUP BY ano ORDER BY total DESC
- Liste os anos que tiveram mais de 3 publicações de livros.
  - SELECT ano FROM livro GROUP BY ano HAVING COUNT(*) > 3
- Liste os anos em que a editora Dom Quixote publicou apenas um livro.
  - SELECT ano FROM livro WHERE editora = “Dom Quixote”
    GROUP BY ano HAVING COUNT(*) = 1

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Subquery

- Uma subquery é um comando SELECT dentro de um comando SELECT.
- Muitas situações só apenas podem ser resolvidas através de subquerys.

- Exemplo de uma estrutura:

```
SELECT coluna
FROM tabela
WHERE coluna = (
        SELECT coluna
        FROM tabela
        WHERE coluna=valor)
ORDER BY coluna DESC
```

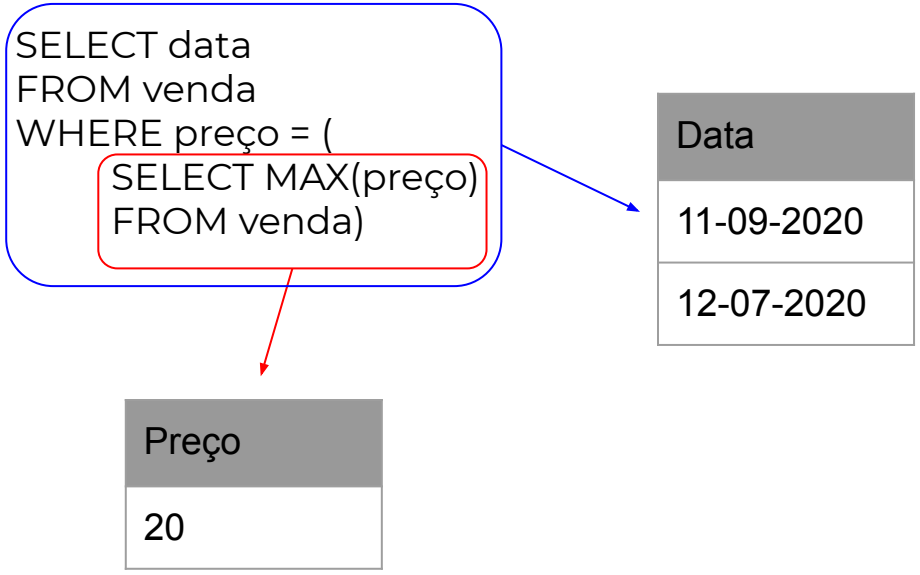
iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Subquery

- Saber a data de venda do produto mais caro.

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20



Subquery

A subquery é particularmente útil quando é necessário efectuar cálculos auxiliares ou obter informação a partir de um conjunto de registos previamente seleccionado

- i.e. fazer a consulta em análises independentes

Localização: varia em função da necessidade na Query Principal podendo ser na cláusula: SELECT, FROM, WHERE e HAVING

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Subquery - Operadores

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

```
SELECT data
FROM venda
WHERE preço = (
        SELECT MAX(preço)
        FROM venda)
```

Preço
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Livro”
AND preço >= ALL (
        SELECT preço
        FROM venda)
```

Preço
15
20
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Pen”
AND preço < ANY (
        SELECT preço
        FROM venda)
```

Preço
1.5
2
1.5

Data	
12-07-2020	id: 102
11-09-2020	id: 105

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Subquery - Operador =

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

```
SELECT data
FROM venda
WHERE preço = (
        SELECT MAX(preço)
        FROM venda)
```

Preço
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Livro”
AND preço >= ALL (
        SELECT preço
        FROM venda
        WHERE produto = “Livro”)
```

Preço
15
20
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Pen”
AND preço < ANY (
        SELECT preço
        FROM venda
        WHERE produto = “Livro”)
```

Preço
1.5
2
1.5

Data	
12-07-2020	id: 102
11-09-2020	id: 105

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Subquery - Operador ALL

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

```
SELECT data
FROM venda
WHERE preço = (
        SELECT MAX(preço)
        FROM venda)
```

Preço
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = "Livro"
AND preço >= ALL (
        SELECT preço
        FROM venda
        WHERE produto="Livro")
```

Preço
15
20
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = "Pen"
AND preço < ANY (
        SELECT preço
        FROM venda
        WHERE produto = "Livro")
```

Preço
1.5
2
1.5

Data	
12-07-2020	id: 102
11-09-2020	id: 105

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Subquery - Operador ANY

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

```
SELECT data
FROM venda
WHERE preço = (
        SELECT MAX(preço)
        FROM venda)
```

Preço
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Livro”
AND preço >= ALL (
        SELECT preço
        FROM venda
        WHERE produto = “Livro”)
```

Preço
15
20
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Pen”
AND preço < ANY (
        SELECT preço
        FROM venda
        WHERE produto=“Pen”)
```

Preço
1.5
2
1.5

Data	
12-07-2020	id: 102
11-09-2020	id: 105

isctе INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Subquery - Operadores

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

- **=** Um operador de comparação simples para comparar **dois valores**.
- **ALL** e **ANY** Combinado com um operador, faz comparação de um valor com cada valor de um grupo:
  - **ALL** Se todas as comparações forem verdadeiras, então é verdadeiro.
  - **ANY** Se pelo menos uma comparação for verdadeira, então é verdadeiro.

```
SELECT data
FROM venda
WHERE preço = (
        SELECT MAX(preço)
        FROM venda)
```

Preço
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Livro”
AND preço >= ALL (
        SELECT preço
        FROM venda
        WHERE produto = “Livro”)
```

Preço
15
20
20

Data	
11-09-2020	id: 106
12-07-2020	id: 103

```
SELECT data
FROM venda
WHERE produto = “Pen”
AND preço < ANY (
        SELECT preço
        FROM venda
        WHERE produto=“Pen”)
```

Preço
1.5
2
1.5

Data	
12-07-2020	id: 102
11-09-2020	id: 105

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Subquery - Operador IN

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

```
SELECT data
FROM venda
WHERE produto = “Livro”
AND preço IN (
        SELECT preço
        FROM venda
        WHERE produto = “Livro”)
```

Preço
15
20
20

Data	
10-08-2020	id: 101
12-07-2020	id: 103
11-09-2020	id: 106

- **IN** ou **NOT IN** avalia se um valor está ou não está incluído num grupo.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Subquery no HAVING

- Como saber qual o produto com maior volume de vendas?

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

1º Passo:

```
SELECT produto, SUM(quantidade)
FROM venda
GROUP BY produto
```

Produto	Quantidade
Livro	30
Pen	45
Caneta	20

2º Passo:

```
SELECT produto
FROM venda
GROUP BY produto
HAVING SUM(quantidade) >= ALL(
        SELECT SUM(quantidade)
        FROM venda
        GROUP BY produto)
```

Produto
Pen

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Subquery no SELECT

- Como saber qual o maior volume de vendas de um produto?

Tabela Venda

Id	Produto	Data	Preço	Quantidade
101	Livro	10-08-2020	15	10
102	Pen	12-07-2020	1.5	15
103	Livro	12-07-2020	20	10
104	Pen	10-09-2020	2	10
105	Pen	11-09-2020	1.5	20
106	Livro	11-09-2020	20	10
107	Caneta	14-10-2020	1	20

1º Passo:

```
SELECT produto, SUM(quantidade) AS
total
FROM venda
GROUP BY produto
```

Produto	Total
Livro	30
Pen	45
Caneta	20

2º Passo:

```
SELECT MAX(total)
FROM (
        SELECT produto,
        SUM(quantidade) AS total
        FROM venda
        GROUP BY produto) AS table
```

Total
45

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

FROM com múltiplas tabelas

Em uma Query, à frente do FROM, devemos indicar a tabela, ou tabelas, que queremos consultar.

Para distinguir as colunas das várias tabelas, utiliza-se como prefixo o nome da tabela ou uma abreviatura.

O emparelhamento entre essas tabelas pode ser feito através da relação chave primária/estrangeira.

```
SELECT <coluna, coluna, ...>
FROM <tabela, tabela, ...>
```

```
SELECT e.nome, n.valor
FROM estudante e, nota n
WHERE e.id_estudante = n.id_estudante
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

FROM com múltiplas tabelas

Tabela Professor

Numero	Nome	Curso
101	Ricardo Ribeiro	Informática
102	Rui Marinheiro	Redes
103	Fernando Batista	Economia

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

```
SELECT e.nome AS Estudante, p.nome AS professor
FROM estudante e, professor p
WHERE e.curso = p.curso
```

O que é que esta Query devolve?

Estudante	Professor
António Brito	Fernando Batista
Daniel Fernandes	Ricardo Ribeiro
Pedro Romano	Ricardo Ribeiro

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

UNION & JOIN
(DML)

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

UNION

Serve para unir os outputs de dois comandos SQL (juntar duas tabelas)

- Ex: Queremos unir estas duas tabelas. Como fazemos?

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

Tabela Professor

Numero	Nome	Curso
101	Ricardo Ribeiro	Informática
102	Rui Marinheiro	Marketing
103	Fernando Batista	Economia

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

UNION

Tabela Professor

Numero	Nome	Curso
101	Ricardo Ribeiro	Informática
102	Rui Marinheiro	Redes
103	Fernando Batista	Economia

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

```
SELECT curso
FROM estudante
UNION
SELECT curso
FROM professor
```

Curso
Economia
Informática
Marketing
Sociologia
História
Null
Redes

- As consultas unidas desta forma devem ter o mesmo **número de colunas**, bem como o **tipo de cada coluna** deve ser o mesmo.
- **UNION** elimina valores duplicados

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

UNION

Tabela Professor

Numero	Nome	Curso
101	Ricardo Ribeiro	Informática
102	Rui Marinheiro	Redes
103	Fernando Batista	Economia

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel Fernandes	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Jorge Rafael	História
20006	José Serro	Null
20007	Pedro Romano	Informática

```
SELECT curso
FROM estudante
UNION ALL
SELECT curso
FROM professor
```

Curso
Economia
Informática
Marketing
Sociologia
História
Null
Informática
Informática
Redes
Economia

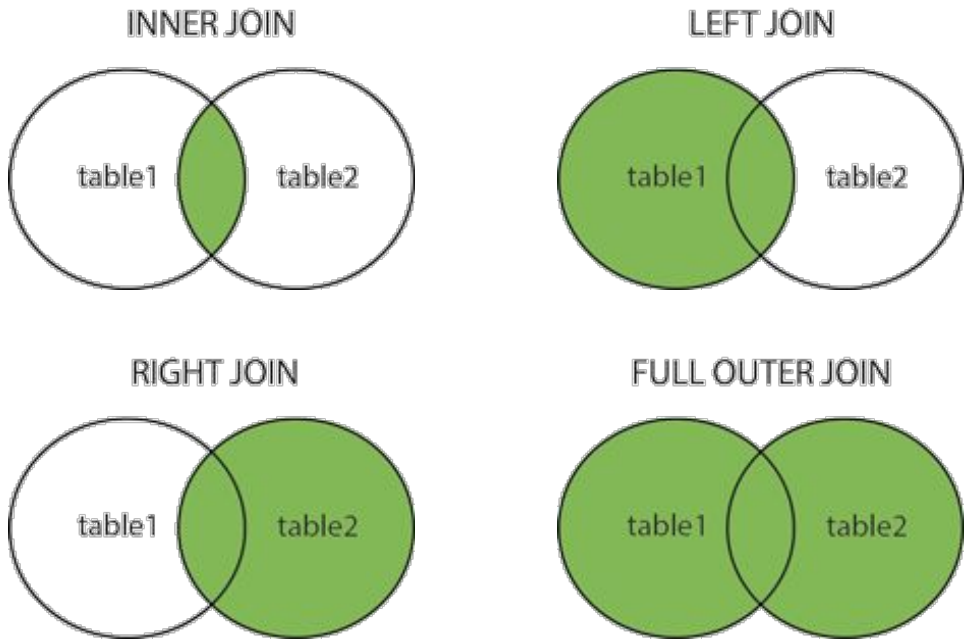
- As consultas unidas desta forma devem ter o mesmo **número de colunas**, bem como o **tipo de cada coluna** deve ser o mesmo.
- **UNION** elimina valores duplicados.
- **UNION ALL** não elimina os valores duplicados.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

JOIN

A cláusula JOIN serve para combinar linhas de duas ou mais tabelas tendo como base uma coluna relacionada entre elas.

- **(INNER) JOIN**
- **LEFT (OUTER) JOIN**
- **RIGHT (OUTER) JOIN**
- **FULL (OUTER) JOIN**



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

(INNER) JOIN

Tabela Estudante

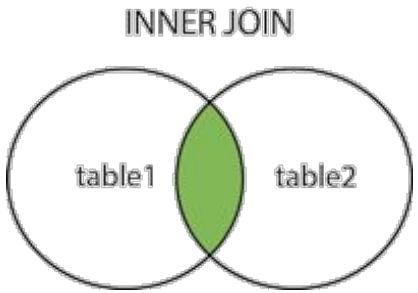
Numero	Nome	Id_Curso
20001	António Brito	1
20002	Daniel Fernandes	4
20003	João Antão	6
20004	João Pavia	3
20005	Jorge Rafael	2
20006	José Serro	Null
20007	Pedro Romano	4

Tabela Curso

Id_Curso	Nome
1	Economia
2	História
3	Sociologia
4	Informática
5	Arquitetura
6	Marketing
7	Gestão

Nome	Curso
António Brito	Economia
Daniel Fernandes	Informática
João Antão	Marketing
João Pavia	Sociologia
Jorge Rafael	História
Pedro Romano	Informática

```
SELECT e.nome AS Nome, c.nome AS Curso
FROM estudante e
JOIN curso c ON e.id_curso = c.id_curso
```



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

LEFT JOIN

Tabela Estudante

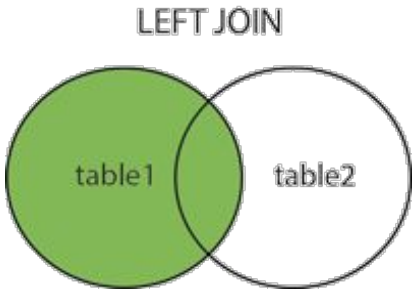
Numero	Nome	Id_Curso
20001	António Brito	1
20002	Daniel Fernandes	4
20003	João Antão	6
20004	João Pavia	3
20005	Jorge Rafael	2
20006	José Serro	Null
20007	Pedro Romano	4

Tabela Curso

Id_Curso	Nome
1	Economia
2	História
3	Sociologia
4	Informática
5	Arquitetura
6	Marketing
7	Gestão

Nome	Curso
António Brito	Economia
Daniel Fernandes	Informática
João Antão	Marketing
João Pavia	Sociologia
Jorge Rafael	História
José Serro	Null
Pedro Romano	Informática

```
SELECT e.nome AS Nome, c.nome AS Curso
FROM estudante e
LEFT JOIN curso c ON e.id_curso = c.id_curso
```



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

RIGHT JOIN

Tabela Estudante

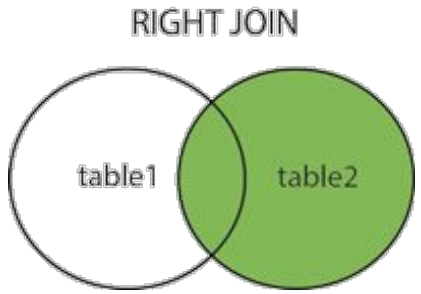
Numero	Nome	Id_Curso
20001	António Brito	1
20002	Daniel Fernandes	4
20003	João Antão	6
20004	João Pavia	3
20005	Jorge Rafael	2
20006	José Serro	Null
20007	Pedro Romano	4

Tabela Curso

Id_Curso	Nome
1	Economia
2	História
3	Sociologia
4	Informática
5	Arquitetura
6	Marketing
7	Gestão

Nome	Curso
António Brito	Economia
Daniel Fernandes	Informática
João Antão	Marketing
João Pavia	Sociologia
Jorge Rafael	História
Pedro Romano	Informática
Null	Arquitetura
Null	Gestão

```
SELECT e.nome AS Nome, c.nome AS Curso
FROM estudante e
RIGHT JOIN curso c ON e.id_curso = c.id_curso
```



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

FULL JOIN

Tabela Estudante

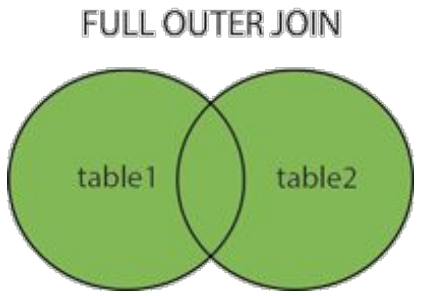
Numero	Nome	Id_Curso
20001	António Brito	1
20002	Daniel Fernandes	4
20003	João Antão	6
20004	João Pavia	3
20005	Jorge Rafael	2
20006	José Serro	Null
20007	Pedro Romano	4

Tabela Curso

Id_Curso	Nome
1	Economia
2	História
3	Sociologia
4	Informática
5	Arquitetura
6	Marketing
7	Gestão

Nome	Curso
António Brito	Economia
Daniel Fernandes	Informática
João Antão	Marketing
João Pavia	Sociologia
Jorge Rafael	História
José Serro	Null
Pedro Romano	Informática
Null	Arquitetura
Null	Gestão

```
SELECT e.nome AS Nome, c.nome AS Curso
FROM estudante e
FULL JOIN curso c ON e.id_curso = c.id_curso
```



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

ON UPDATE/ON DELETE

Para situações em que atualizamos ou apagamos chaves primárias, e estas têm uma relação com uma outra tabela, o que acontece?

Tabela Estudante

Numero	Nome	Id_Curso
20001	António Brito	1
20002	Daniel Fernandes	4
20003	João Antão	6
20004	João Pavia	3
20005	Jorge Rafael	2
20006	José Serro	Null
20007	Pedro Romano	4

?

Tabela Curso

Id_Curso	Nome
1	Economia
2	História
3	Sociologia
4	Informática
5	Arquitetura
6	rketing
7	Gestão

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

ON UPDATE/ON DELETE

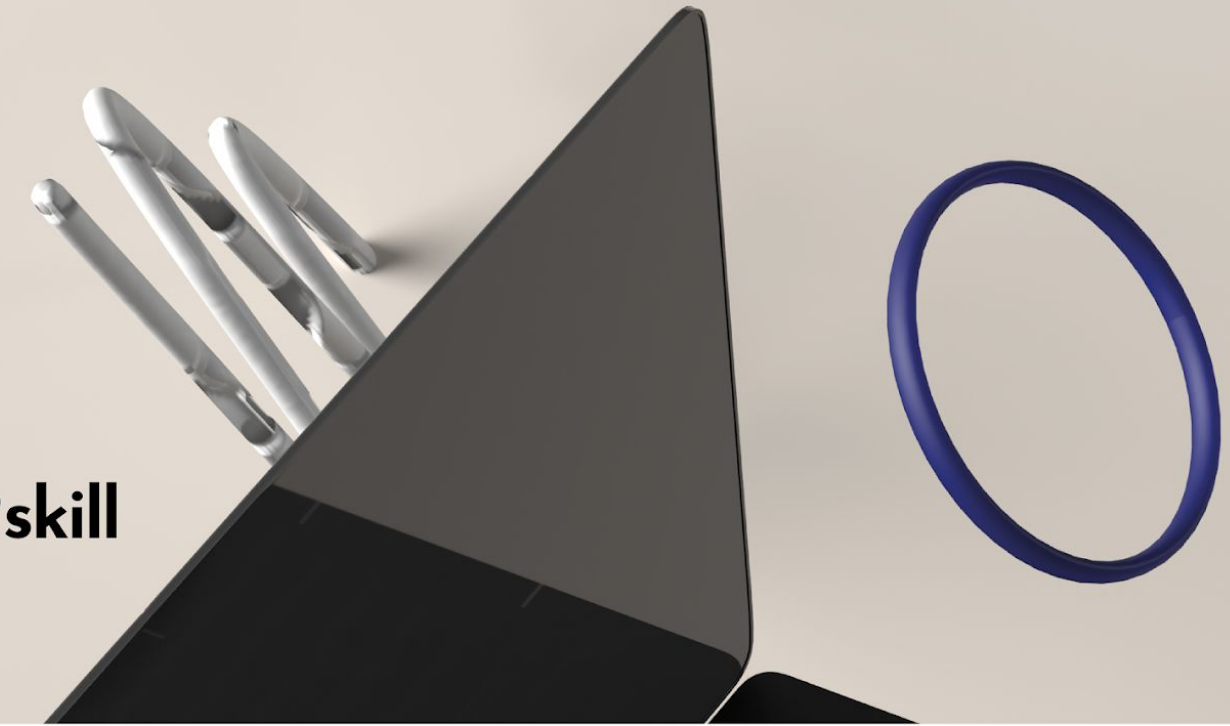
Podemos configurar estas situações de 4 maneiras:

- **RESTRICT** - Impede que os dados do pai sejam atualizados/eliminados.
- **CASCADE** - Os dados do filho são atualizados/eliminados quando os do pai são atualizados/eliminados.
- **SET NULL** - Os dados do filho são definidos como NULL quando os do pai são atualizados/eliminados.
- **NO ACTION** - Nenhuma ação ocorre no filho quando os dados do pai são atualizados/eliminados.

Cada uma destas opções é selecionada para a situação de UPDATE e de DELETE, podendo ser diferentes.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill