



iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA



emprego
digital

Módulo 5: Bases de Dados

Aula 8

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



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 queries:
 - WHERE
- Ordenar queries:
 - ORDER BY

**Hoje vamos continuar a
aprender mais sobre SQL!**

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
- **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

Funções de Agregação

Tabela “Estudante”

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel F.	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

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?

GROUP BY

○ **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 Filipe	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


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 Filipe	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

O HAVING seleciona grupos, mas...

**Na ausência de um GROUP BY,
comporta-se como WHERE**

WHERE vs HAVING

WHERE	HAVING
A cláusula WHERE tem como finalidade seleccionar registos (linhas).	A cláusula HAVING tem como finalidade seleccionar grupos que obedecem 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.	

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.

Subquery

- Uma subquery é um comando SELECT dentro de um comando SELECT.
- Muitas situações só apenas podem ser resolvidas através de subqueries.
- Exemplo de uma subquery:

```
SELECT coluna  
FROM tabela  
WHERE coluna = (  
    SELECT coluna2  
    FROM tabela2  
    WHERE coluna2=valor)  
ORDER BY coluna DESC
```

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

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

Data
11-09-2020
12-07-2020

Preço
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

Onde podemos usar?

Varia em função da necessidade na query principal podendo ser na cláusula: SELECT, FROM, WHERE e HAVING

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
    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

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="Pen")
```

Preço

1.5

2

1.5

Data

12-07-2020

id: 102

11-09-2020

id: 105

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="Pen")
```

Preço

1.5

2

1.5

Data

12-07-2020

id: 102

11-09-2020

id: 105

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

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
12-07-2020
11-09-2020

id: 101

id: 103

id: 106

Subquery - Operadores - **Resumo**

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

Subquery no **HAVING**

Como saber 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

Subquery no **FROM**

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)
FROM venda
GROUP BY produto
```

Produto	Quantidade
Livro	30
Pen	45
Caneta	20

2º Passo:

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

Total
45

FROM com múltiplas tabelas

- Numa query, devemos indicar a tabela, ou tabelas, que queremos consultar à frente do FROM.
- 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
```

FROM com múltiplas tabelas

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 Silva	Informática

Tabela Professor

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

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

O que devolve esta Query?

FROM com múltiplas tabelas

Tabela Estudante

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

Tabela Professor

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

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

O que devolve esta Query?

Resultado Query

Estudante	Professor
António Brito	Fernando Batista
Daniel F.	Ricardo Ribeiro
Pedro Silva	Ricardo Ribeiro

UNION & JOIN

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 F.	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Pedro Romano	Informática

Tabela Professor

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

UNION

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel F.	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Pedro Romano	Informática

Tabela Professor

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

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



Curso
Economia
Informática
Marketing
Sociologia
História

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

UNION ALL

Tabela Estudante

Numero	Nome	Curso
20001	António Brito	Economia
20002	Daniel F.	Informática
20003	João Antão	Marketing
20004	João Pavia	Sociologia
20005	Pedro Romano	Informática

Tabela Professor

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

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

- O **UNION ALL** não elimina valores duplicados

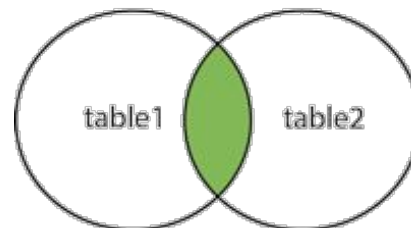
Curso
Economia
Informática
Marketing
Sociologia
Informática
Informática
Marketing
História

JOIN

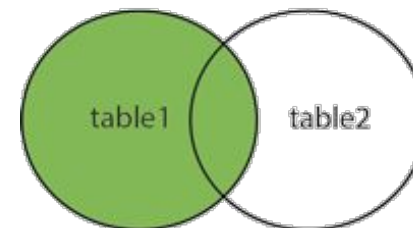
A cláusula JOIN serve para combinar linhas de duas ou mais tabelas tendo como base uma coluna relacionada entre elas.

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

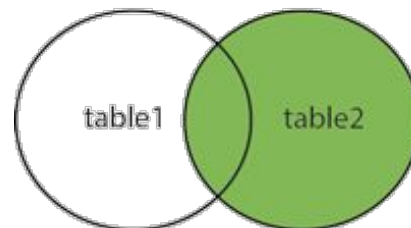
INNER JOIN



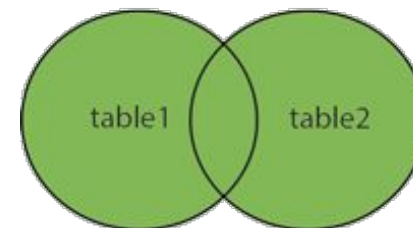
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



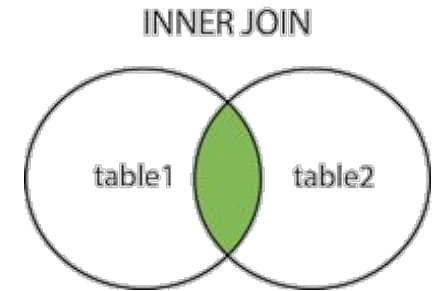
INNER JOIN

Tabela Estudante

Numero	Nome	ID Curso
20001	António Brito	1
20002	Daniel F.	1
20003	João Antão	2
20004	João Pavia	NULL
20005	Pedro Romano	1

Tabela Curso

ID Curso	Nome
1	Economia
2	História
3	Sociologia



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



Nome	Curso
Antonio Brito	Economia
Daniel F.	Economia
João Antão	História
Pedro Romano	Economia

LEFT JOIN

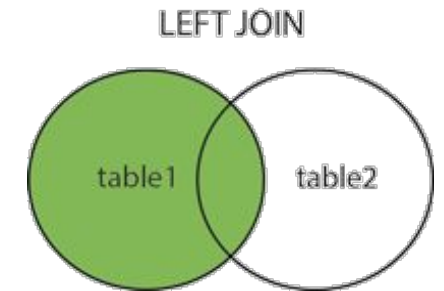


Tabela Estudante

Numero	Nome	ID Curso
20001	António Brito	1
20002	Daniel F.	1
20003	João Antão	2
20004	João Pavia	NULL
20005	Pedro Romano	1

Tabela Curso

ID Curso	Nome
1	Economia
2	História
3	Sociologia

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

Nome	Curso
Antonio Brito	Economia
Daniel F.	Economia
João Antão	História
João Pavia	NULL
Pedro Romano	Economia

RIGHT JOIN

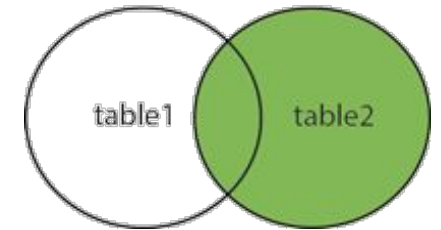
Tabela Estudante

Numero	Nome	ID Curso
20001	António Brito	1
20002	Daniel F.	1
20003	João Antão	2
20004	João Pavia	NULL
20005	Pedro Romano	1

Tabela Curso

ID Curso	Nome
1	Economia
2	História
3	Sociologia

RIGHT JOIN



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



Nome	Curso
Antonio Brito	Economia
Daniel F.	Economia
João Antão	História
Pedro Romano	Economia
NULL	Sociologia

FULL JOIN

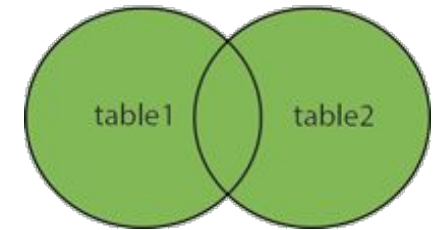
Tabela Estudante

Numero	Nome	ID Curso
20001	António Brito	1
20002	Daniel F.	1
20003	João Antão	2
20004	João Pavia	NULL
20005	Pedro Romano	1

Tabela Curso

ID Curso	Nome
1	Economia
2	História
3	Sociologia

FULL OUTER JOIN



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



Nome	Curso
Antonio Brito	Economia
Daniel F.	Economia
João Antão	História
João Pavia	NULL
Pedro Romano	Economia
NULL	Sociologia

**O que acontece quando apagamos
um registo que era referenciado
noutra tabela?**

Temos várias hipóteses...

Regras de Integridade

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 F.	1
20003	João Antão	2 ?
20004	João Pavia	NULL
20005	Pedro Romano	1

Tabela Curso

Numero	Nome
1	Economia
2	História
3	Sociologia

Regras de Integridade

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 seleccionada para a situação de **UPDATE** e de **DELETE**, podendo ser diferentes.

O futuro profissional começa aqui

iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA



emprego
digital



UPskill