



# Data Query Language(DQL)

Linguagem de Consulta de Dados  
DBII



# SQL

```
graph TD; SQL[SQL] -.-> SUBCONJUNTOS_SQL[SUBCONJUNTOS SQL]; SUBCONJUNTOS_SQL -.-> DQL[DQL]; SUBCONJUNTOS_SQL -.-> DML[DML]; SUBCONJUNTOS_SQL -.-> DDL[DDL]; SUBCONJUNTOS_SQL -.-> DCL[DCL]; SUBCONJUNTOS_SQL -.-> DTL[DTL]; DQL -.-> SELECT[SELECT]; DML -.-> INSERT_UPDATE_DELETE[INSERT<br/>UPDATE<br/>DELETE]; DDL -.-> CREATE_ALTER_DROP[CREATE<br/>ALTER<br/>DROP]; DCL -.-> GRANT_REVOKE[GRANT<br/>REVOKE]; DTL -.-> BEGIN_COMMIT_ROLLBACK[BEGIN<br/>COMMIT<br/>ROLLBACK];
```

SUBCONJUNTOS SQL

**DQL**

**DML**

**DDL**

**DCL**

**DTL**

COMANDOS SQL

SELECT

INSERT  
UPDATE  
DELETE

CREATE  
ALTER  
DROP

GRANT  
REVOKE

BEGIN  
COMMIT  
ROLLBACK



# SELECT

---

Utilizado para consultar dados no banco.

```
select nome
from tb_escolas_de_TI as escolas_TI
where (escolas_TI.qualidade LIKE "EXCELENTE")
      AND (escolas_TI.localizacao LIKE "PORTO ALEGRE")
```

# SELECT \* FROM nome\_tabela;

---

Seleciona todas as  
colunas e registros  
da tabela

```
1  
2 • SELECT * FROM alunos;
```

\*

Indica que  
queremos  
todas as colunas



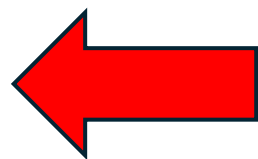
Result Grid



Filter Rows:

	id	nome	idade	curso
▶	1	Ana Paula	17	Informática
	2	Bruno Silva	18	Administração
	3	Carlos Oliveira	17	Informática
	4	Daniela Lima	19	Contabilidade
	5	Eduardo Dias	18	Informática
	6	Fernanda Souza	17	Administração
	7	Gustavo Lima	20	Contabilidade

```
2  ●  SELECT
3      nome,
4      idade
5  FROM
6      alunos;
```



Podemos selecionar uma ou mais colunas

<		
Result Grid		
Filter Rows:		
	nome	idade
▶	Ana Paula	17
	Bruno Silva	18
	Carlos Oliveira	17
	Daniela Lima	19
	Eduardo Dias	18
	Fernanda Souza	17
	Gustavo Lima	20
	Helena Castro	16
	Igor Ramos	19
	Julia Martins	17
	Kaique Alves	18
	Lucas T.	17



---

## Boas Práticas

Evite fazer consultas e/ou DML sem a cláusula WHERE, você pode afetar todos os registros da tabela, o que pode gerar erros graves ou resultados inesperados.



```
1
2 • SELECT *
3 FROM alunos
4 WHERE idade >= 18;
```

< Result Grid   Filter Rows:

	id	nome	idade	curso
▶	2	Bruno Silva	18	Administração
	4	Daniela Lima	19	Contabilidade
	5	Eduardo Dias	18	Informática
	7	Gustavo Lima	20	Contabilidade
	9	Igor Ramos	19	Informática
	11	Kaique Alves	18	Informática
	13	Marcos Nunes	19	Administração
	14	Natália Mendes	18	Informática
	16	Paula Fernandes	20	Administração
	19	Thiago Costa	18	Informática
	20	Vanessa Ribeiro	19	Administração
	NULL	NULL	NULL	NULL

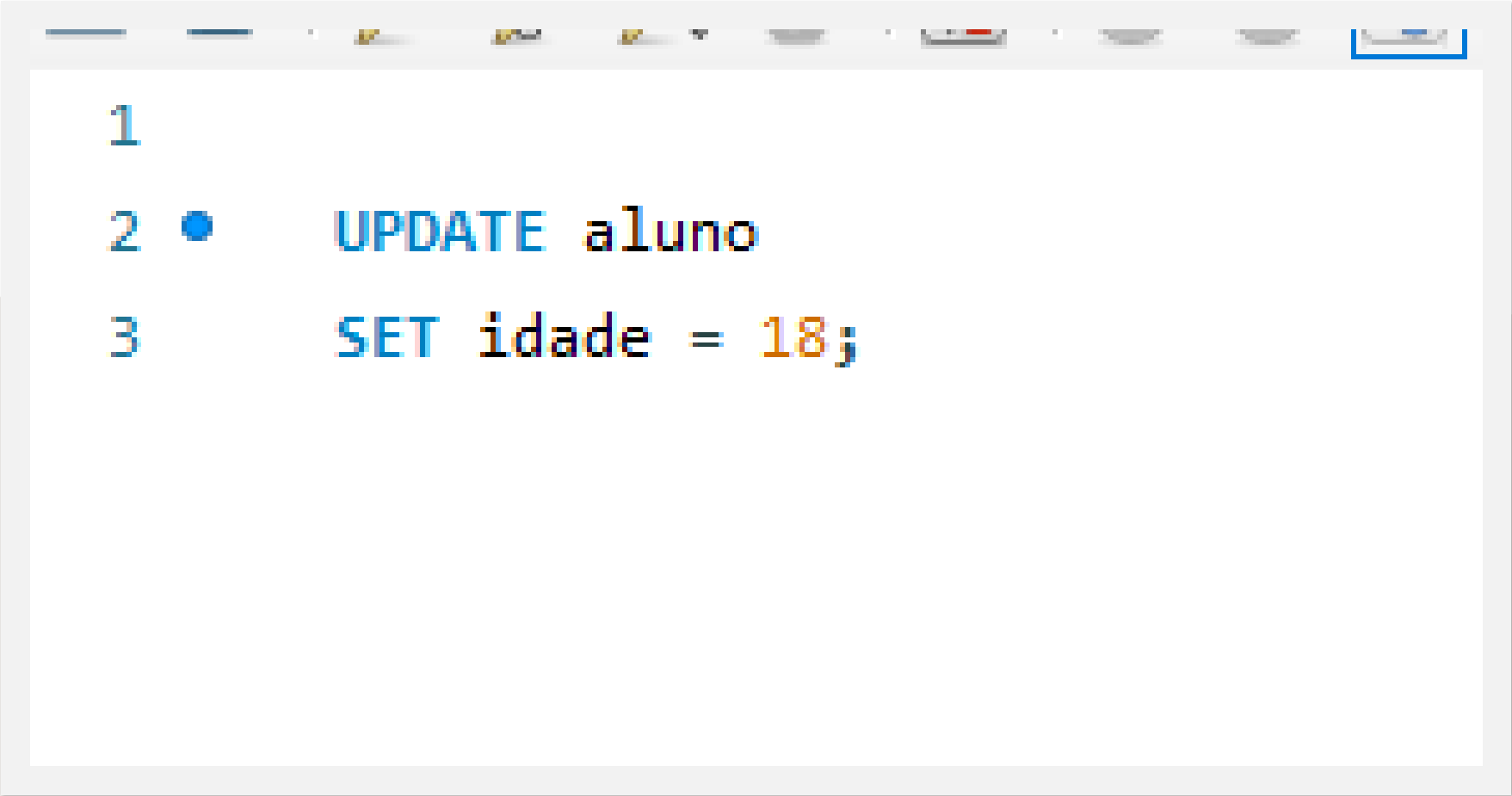
# WHERE

Filtra registros com base em uma condição

Exemplo.:

```
SELECT *
FROM alunos
WHERE idade >= 18;
```

*Filtrar pela coluna **idade**, onde o valor da **idade** seja **maior ou igual a 18**.*



```
1  
2 • UPDATE aluno  
3 SET idade = 18;
```

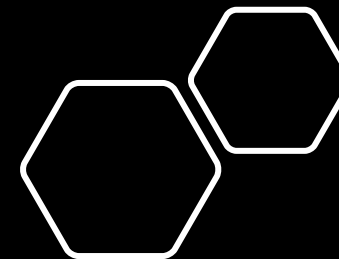
**O que aconteceria?**



**EM CASO DE  
UPDATE SEM WHERE**



**COMPAREÇA  
A SALA DO RH**



# Nunca utilize instrução DML sem o **WHERE**

```
1  
2 • UPDATE aluno  
3   SET idade = 18;
```



X

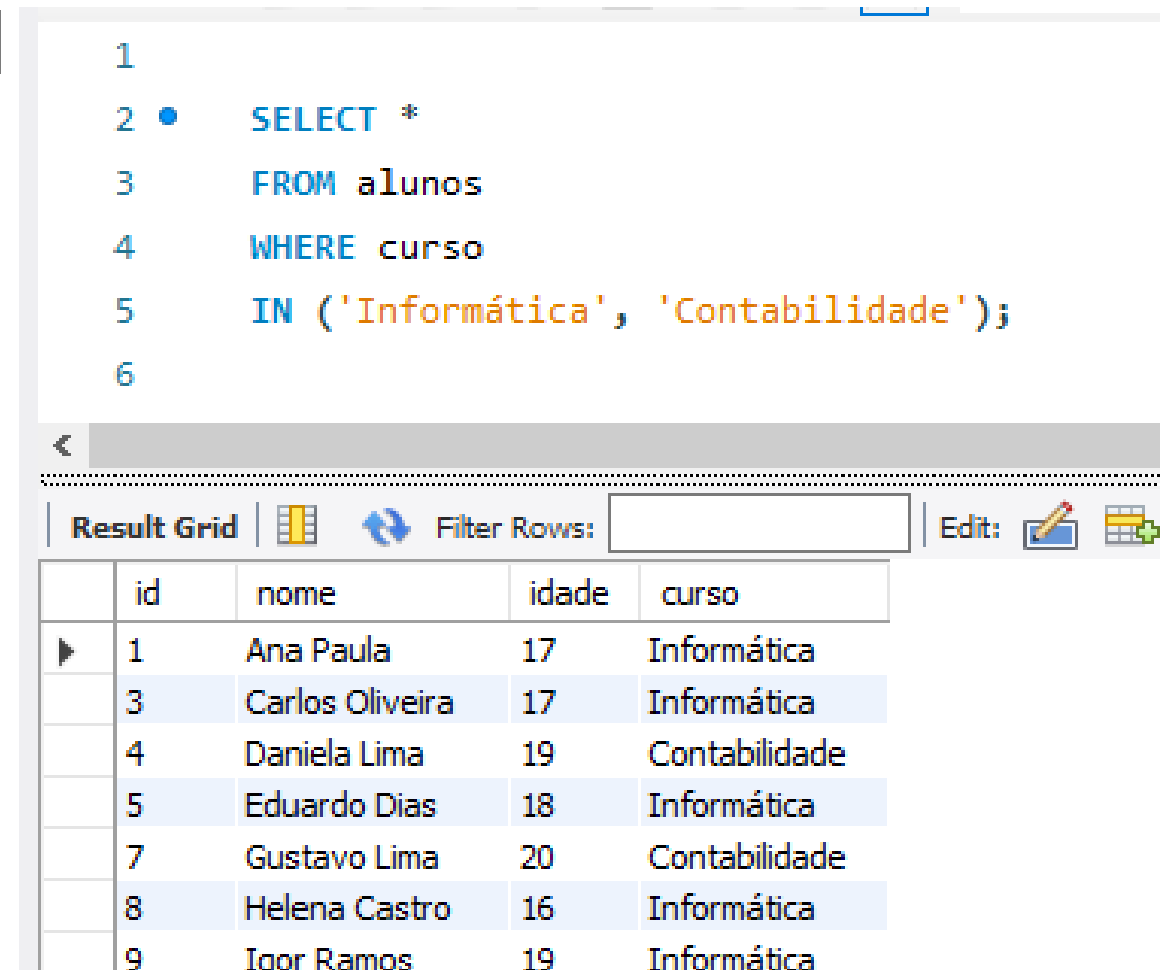
```
1  
2 • UPDATE aluno  
3   SET idade = 18  
4   WHERE id = 2;
```



## IN - Filtrar registros dentro de uma lista

---

```
SELECT *  
FROM alunos  
WHERE curso  
IN ('Informática',  
'Contabilidade');
```



The screenshot shows a SQL query editor with a query and its corresponding results grid. The query is as follows:

```
1  
2 • SELECT *  
3 FROM alunos  
4 WHERE curso  
5 IN ('Informática', 'Contabilidade');  
6
```

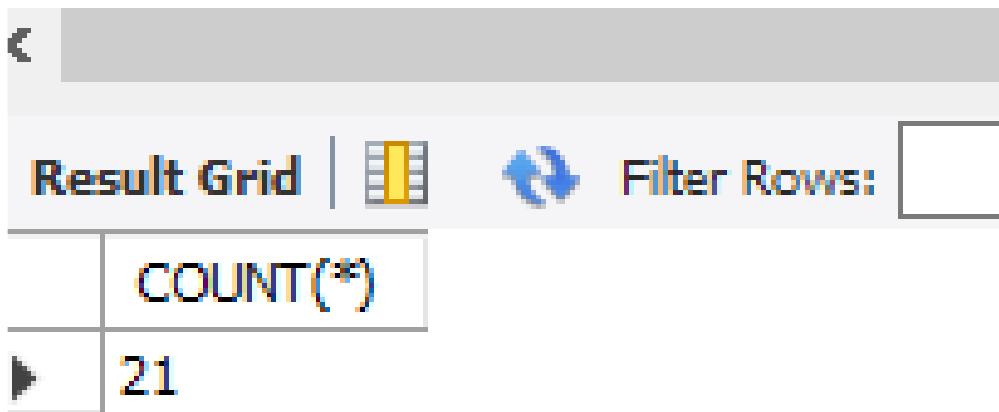
Below the query, the results grid is displayed. It has a toolbar with 'Result Grid', 'Filter Rows', and 'Edit' options. The grid contains the following data:

	id	nome	idade	curso
▶	1	Ana Paula	17	Informática
	3	Carlos Oliveira	17	Informática
	4	Daniela Lima	19	Contabilidade
	5	Eduardo Dias	18	Informática
	7	Gustavo Lima	20	Contabilidade
	8	Helena Castro	16	Informática
	9	Ioor Ramos	19	Informática

# FUNÇÕES AGREGADAS



```
1
2 • SELECT COUNT(*)
3 FROM alunos;
4
```



COUNT(*)
21

# COUNT()

Contabiliza a quantidade de registros, o exemplo abaixo contabiliza todos os registros da tabela alunos.

SELECT **COUNT(\*)** FROM alunos;

- **COUNT(\*)**: conta todas as linhas
- **COUNT(coluna)**: conta apenas os valores não nulos
- **COUNT(DISTINCT coluna)**: conta valores únicos não nulos

# SUM()

Soma os valores de uma coluna numérica

*AS é um apelido...*

```
1 • SELECT SUM(idade) AS soma_idades
2 FROM alunos
3 WHERE curso = 'Informática';
```

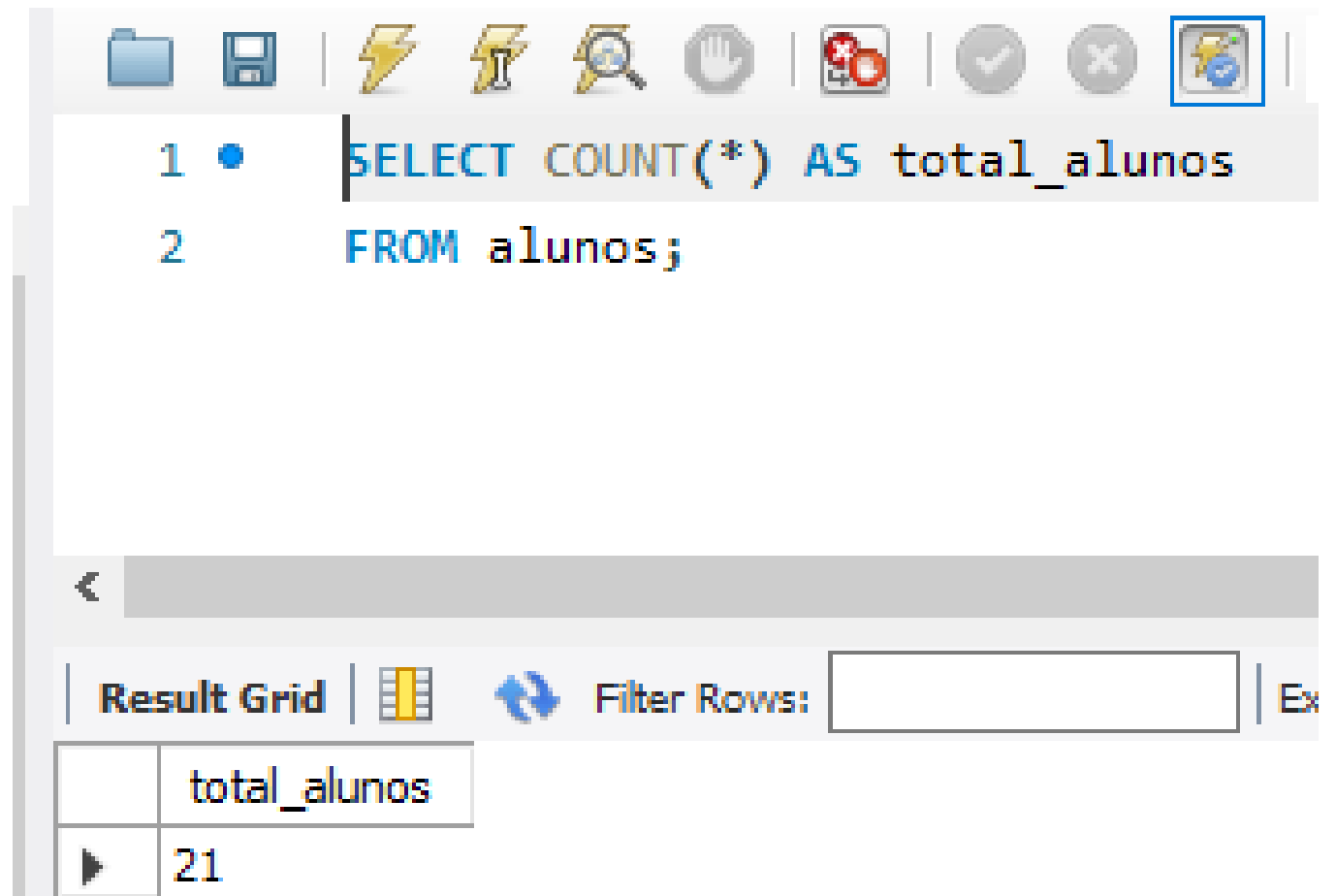
Result Grid

	soma_idades
▶	157

# AS

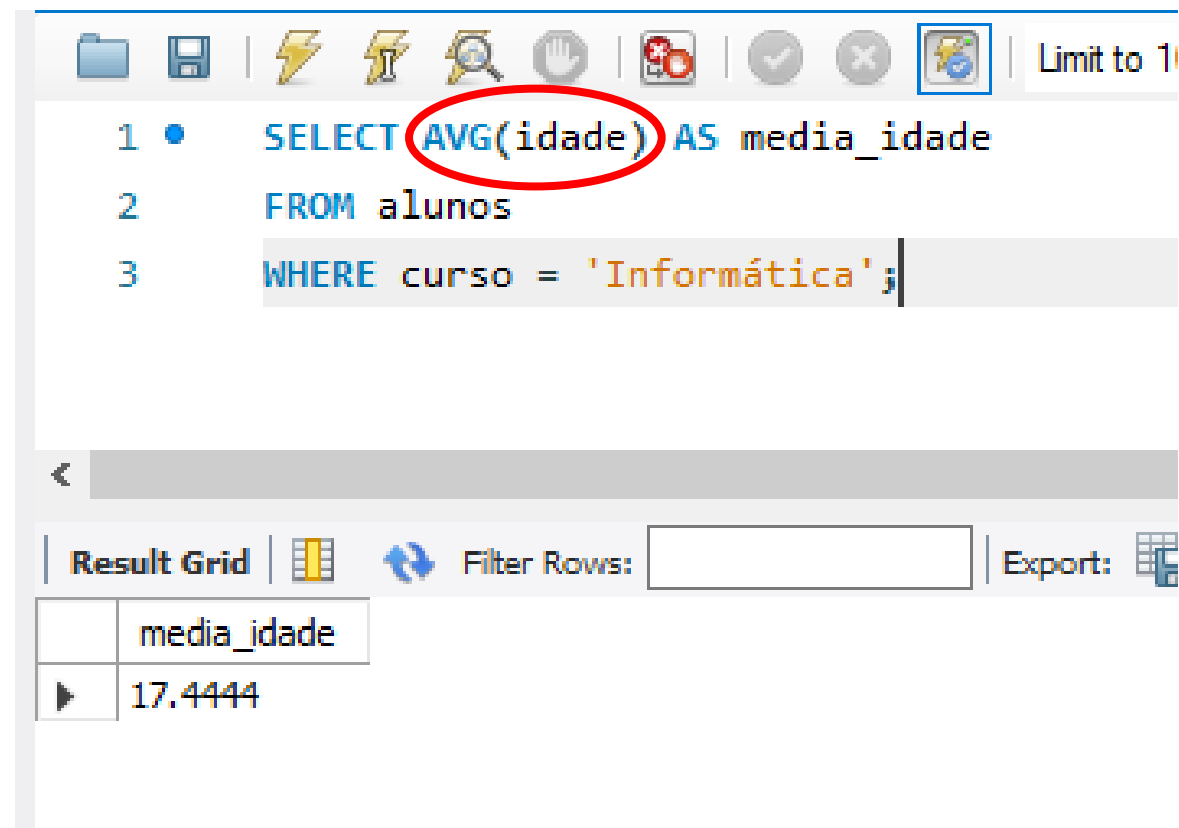
## *Alias / Apelido*

**Apelido temporário** dado a uma **coluna** ou **tabela** em uma consulta



Calcula a média dos  
valores de uma coluna

# AVG



The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 • SELECT AVG(idade) AS media_idade
2 FROM alunos
3 WHERE curso = 'Informática';
```

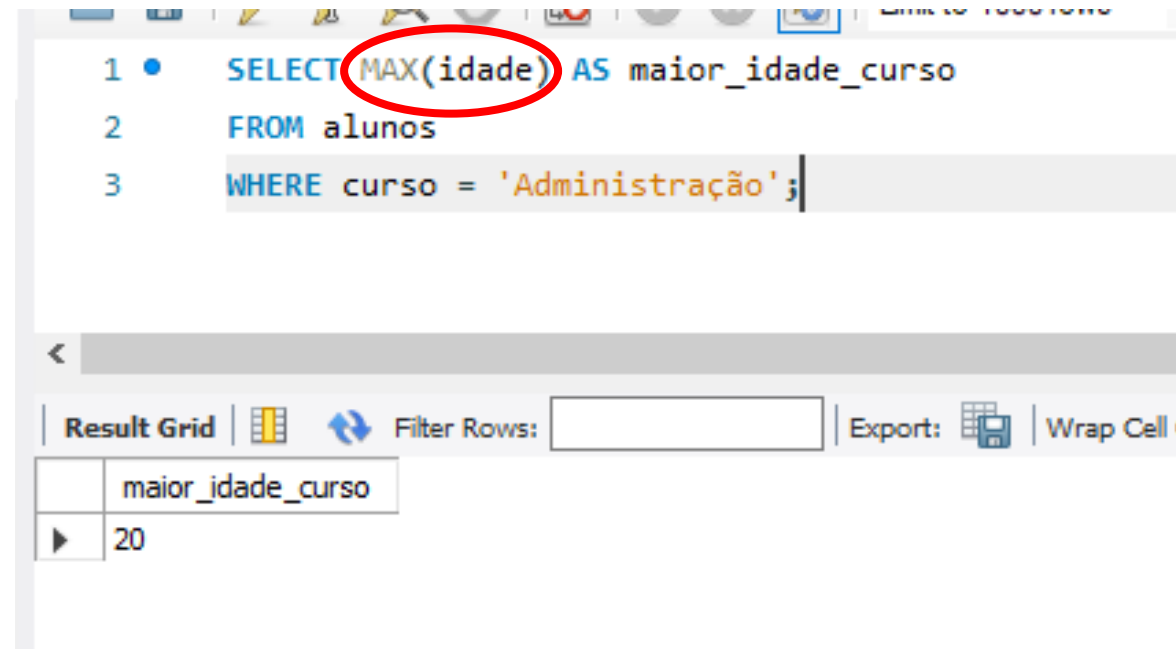
The function `AVG(idade)` is circled in red. Below the query editor, there is a section for the results. It includes a "Result Grid" tab, a "Filter Rows:" input field, and an "Export:" button. The result grid displays the following data:

	media_idade
▶	17.4444



retorna o maior valor de  
uma coluna

# MAX()



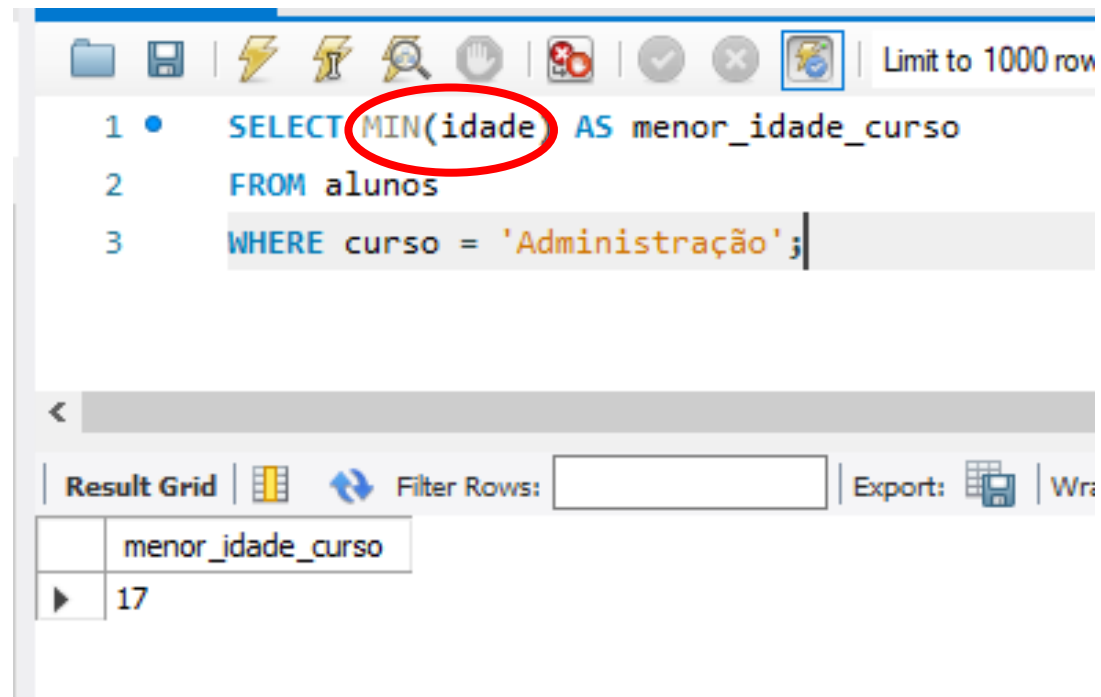
```
1 • SELECT MAX(idade) AS maior_idade_curso
2 FROM alunos
3 WHERE curso = 'Administração';
```

< Result Grid | Filter Rows: | Export: | Wrap Cell

	maior_idade_curso
▶	20

# MIN()

retorna o menor valor de uma coluna



Limit to 1000 rows

```
1 • SELECT MIN(idade) AS menor_idade_curso
2 FROM alunos
3 WHERE curso = 'Administração';
```

Result Grid

	menor_idade_curso
▶	17

```
1
2 • SELECT nome, idade
3 FROM alunos
4 ORDER BY idade DESC;
5
```

	nome	idade
▶	Gustavo Lima	20
	Paula Fernandes	20
	Igor Ramos	19
	Marcos Nunes	19
	Daniela Lima	19
	Vanessa Ribeiro	19
	Eduardo Dias	18

# ORDER BY

Ordena os resultados de uma consulta.

- **ASC**: ordena de forma crescente (padrão)
- **DESC**: ordena de forma decrescente

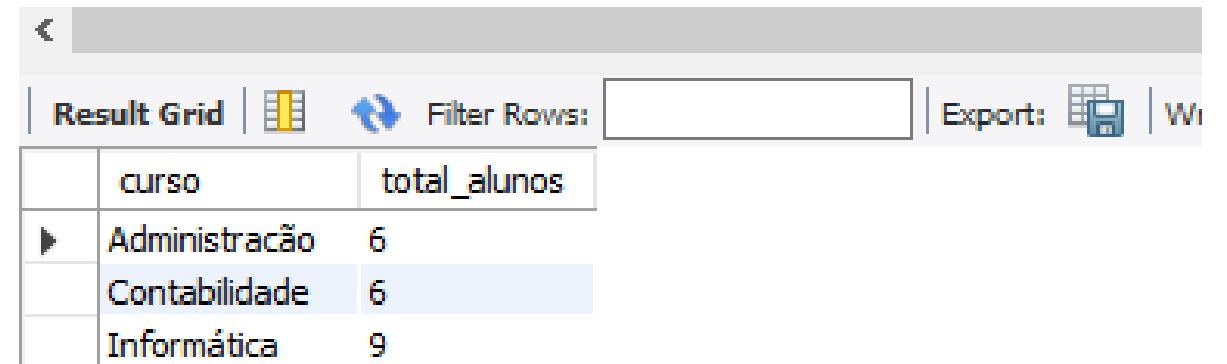
# GROUP BY

---

Agrupa os resultados com base em uma ou mais colunas.

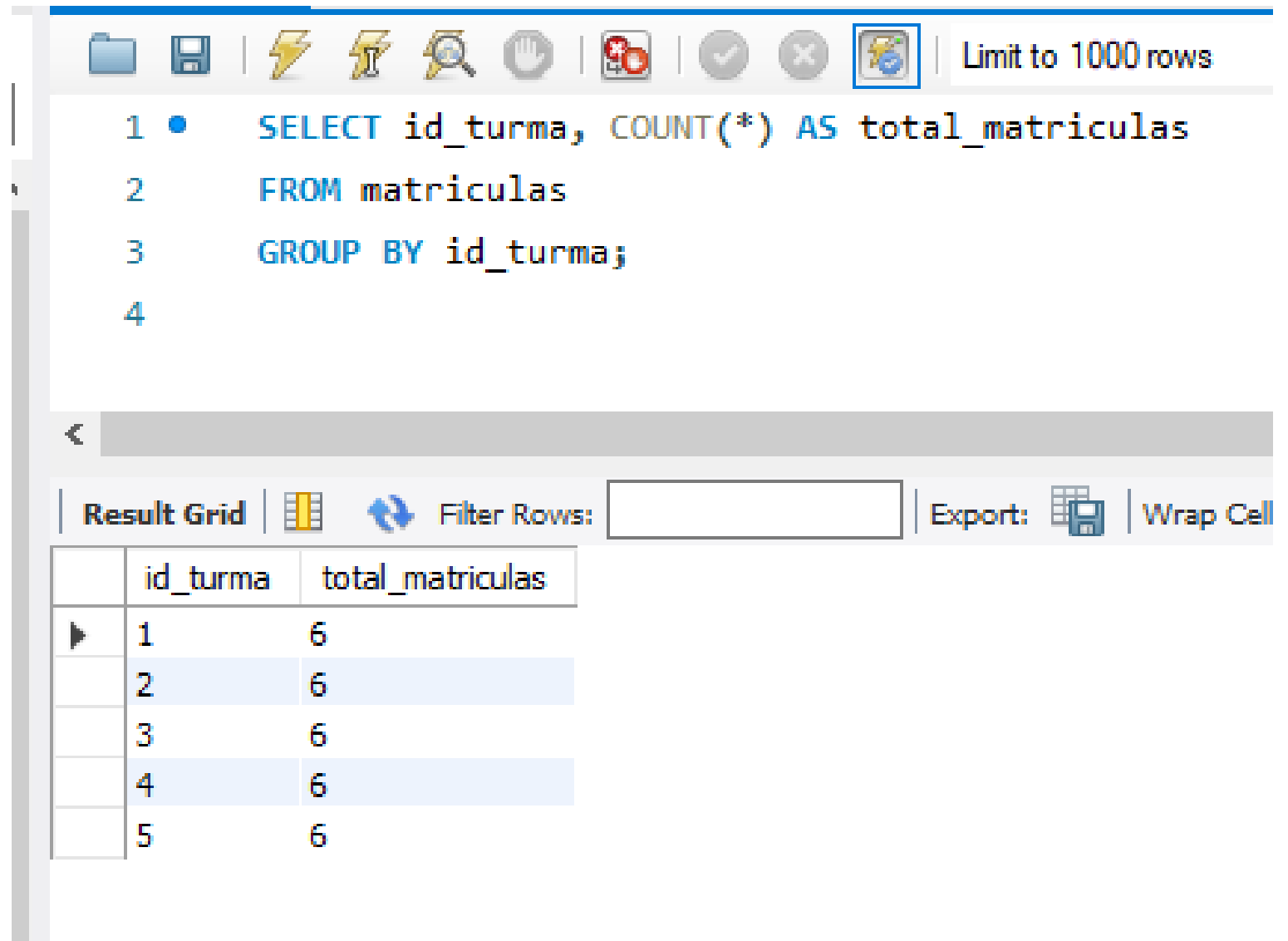
Usado junto com funções agregadas (COUNT, SUM e etc)

```
1 • SELECT curso, COUNT(*) AS total_alunos  
2 FROM alunos  
GROUP BY curso;
```



	curso	total_alunos
▶	Administração	6
	Contabilidade	6
	Informática	9

# Total de matriculas por cursos



The screenshot shows a database query interface. At the top, there is a toolbar with various icons for file operations, execution, and viewing. A text box on the right indicates "Limit to 1000 rows". Below the toolbar, a SQL query is displayed in a text editor:

```
1 • SELECT id_turma, COUNT(*) AS total_matriculas
2 FROM matriculas
3 GROUP BY id_turma;
4
```

Below the query editor, there is a "Result Grid" tab. The grid shows the results of the query, with columns "id\_turma" and "total\_matriculas". The results are as follows:

	id_turma	total_matriculas
▶	1	6
	2	6
	3	6
	4	6
	5	6

At the bottom of the result grid, there are options for "Filter Rows:" (with a search input field), "Export:" (with a download icon), and "Wrap Cell:" (with a checkbox).

# HAVING

## HAVING

Filtra grupos após o uso do  
GROUP BY

Usado em colunas que  
utilizam funções agregadas  
(COUNT, SUM, MAX ...)

X

## WHERE

Filtra os registros antes do  
GROUP BY

Utilizado com colunas  
normais

# HAVING

Funciona parecido com o **WHERE**, só que o where filtra antes de agrupar (**GROUP BY**)

O **HAVING** filtra os resultados depois dos agrupamentos.

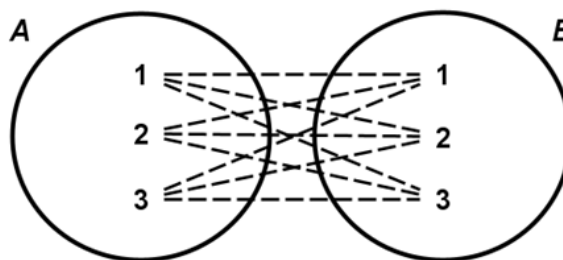
```
1 • SELECT curso, COUNT(*) AS total
2   FROM alunos
3   GROUP BY curso
   → HAVING COUNT(*) > 4;
```

Result Grid			Filter Rows:	
	curso	total		
▶	Administração	6		
	Contabilidade	6		
	Informática	9		

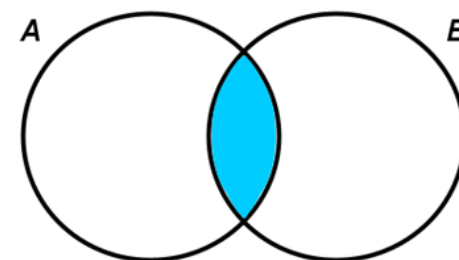
# Junções SQL

Utilizado para **combinar dados de duas ou mais tabelas** em uma única consulta

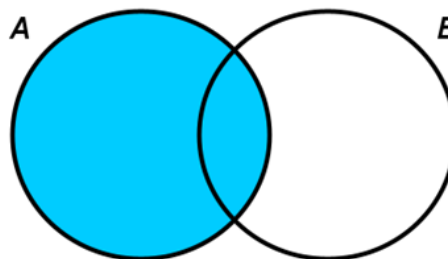
CROSS JOIN



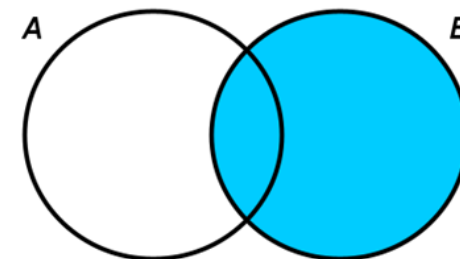
INNER JOIN



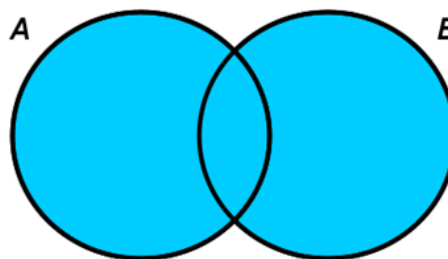
LEFT OUTER JOIN



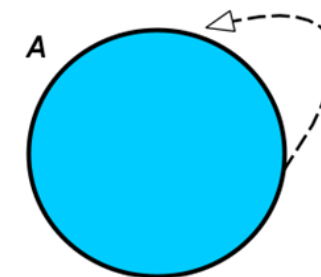
RIGHT OUTER JOIN



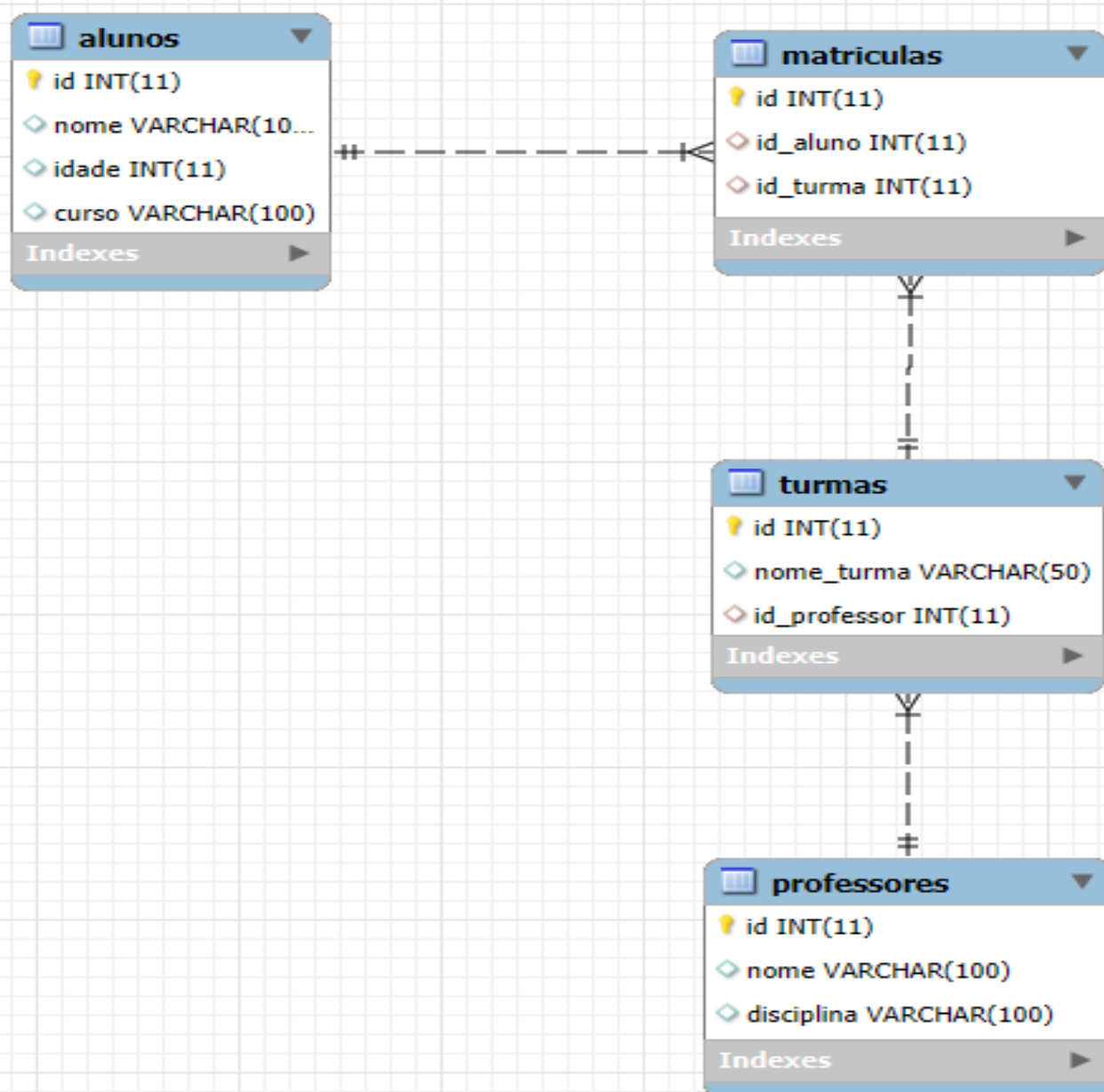
FULL OUTER JOIN



SELF JOIN



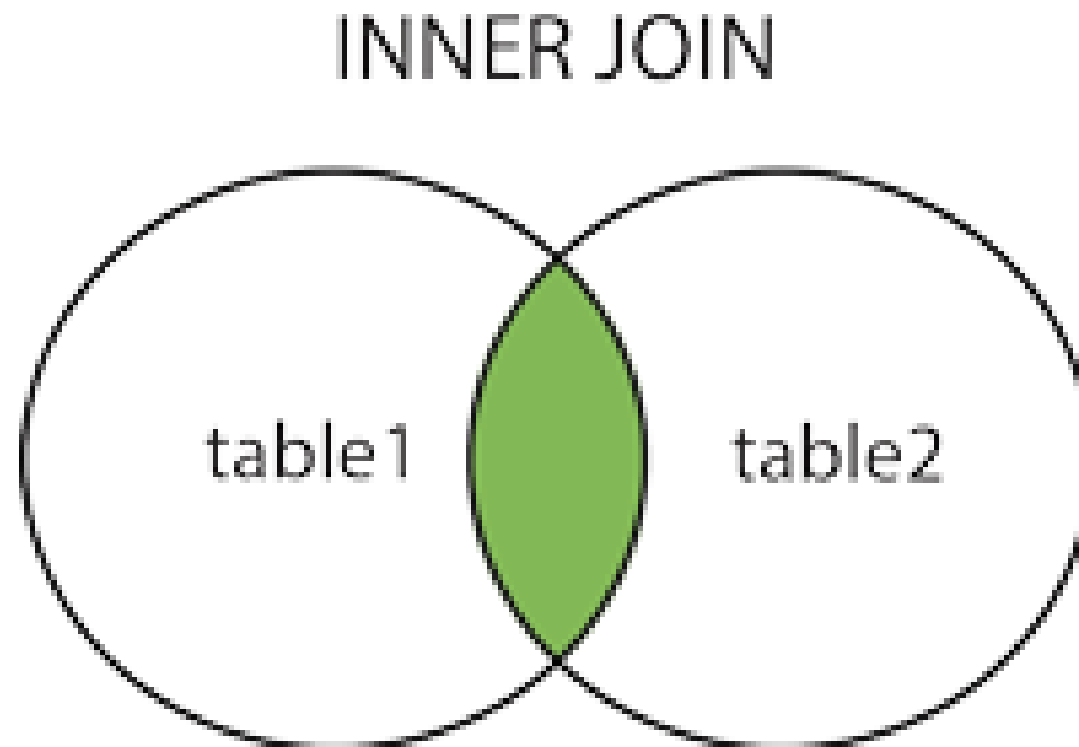




# INNER JOIN

Combina registros de duas ou mais tabelas **somente quando há correspondência** entre elas, com base em uma chave comum (**chave primária e chave estrangeira**).

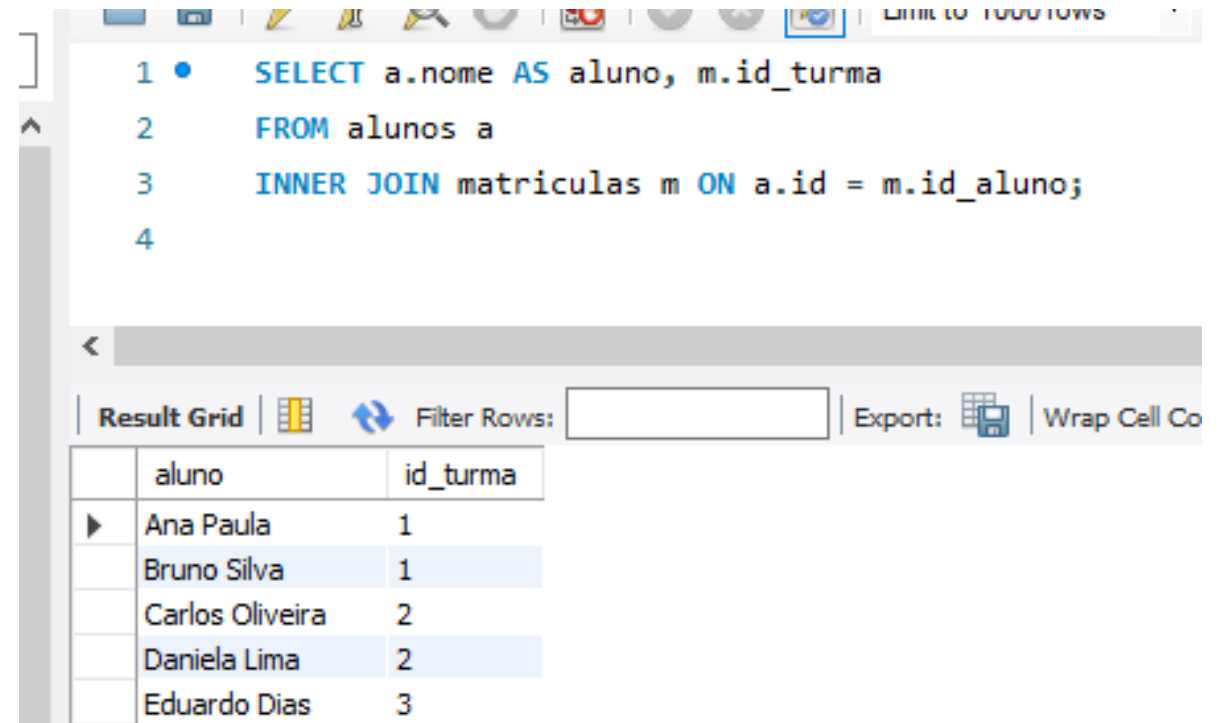
Geralmente utilizamos para exibir **apenas registros que possuem correspondência** em ambas as tabelas.



# INNER JOIN

---

Retornar os *nomes dos alunos* e o *ID das turmas* onde eles estão matriculados



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

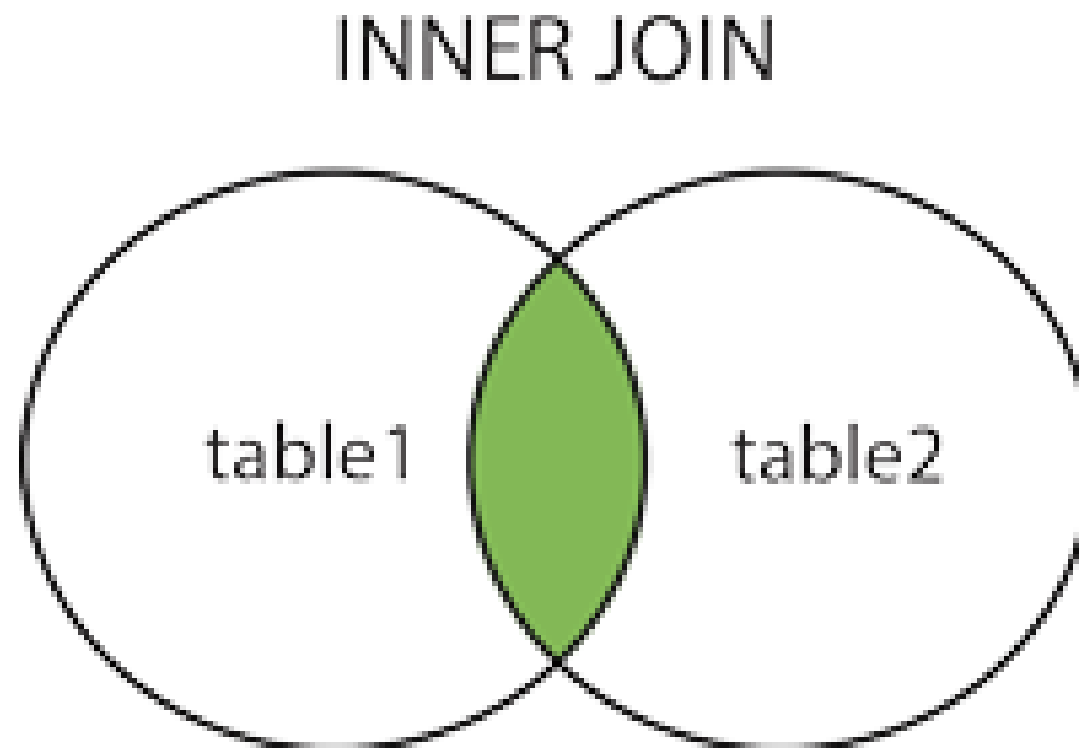
```
1 • SELECT a.nome AS aluno, m.id_turma
2 FROM alunos a
3 INNER JOIN matriculas m ON a.id = m.id_aluno;
4
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Export' button. The results are displayed in a table with two columns: 'aluno' and 'id\_turma'.

	aluno	id_turma
▶	Ana Paula	1
	Bruno Silva	1
	Carlos Oliveira	2
	Daniela Lima	2
	Eduardo Dias	3

# INNER JOIN

Retorna **apenas alunos**  
**que têm matrícula em**  
**alguma turma.**



# INNER JOIN

Agora, ao invés de mostrar o **nome do aluno** e o **ID da turma**...

vamos exibir no lugar do ID o **nome da turma**.

```
1 • SELECT a.nome AS aluno, t.nome_turma
2 FROM alunos a
3 INNER JOIN matriculas m ON a.id = m.id_aluno
4 INNER JOIN turmas t ON t.id = m.id_turma;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	aluno	nome_turma			
▶	Ana Paula	Turma A			
	Bruno Silva	Turma A			
	Kaique Alves	Turma A			
	Paula Fernandes	Turma A			
	William Pinto	Turma A			
	Eduardo Dias	Turma A			
	Carlos Oliveira	Turma B			



**Podemos  
avançar?**

Ou aqui está  
bom?

*Ainda tem Left,  
Right, Full Outer  
Join...*

# Desafio 2

Exibir o **nome do curso** e a **quantidade de matérias** que ele possui

- Count
- Group by



Tem muitas coisas  
que podemos  
aprender...





# Professores

- Cleiton S Dias
- Thiago G Pascotto





## Contatos



**cleitondsd**



**cleitondsd**



**(11) 9 3029-0421**