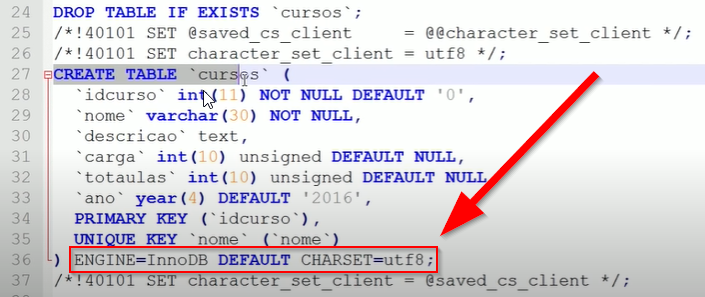
Aula 11

**RELACIONANDO AS TABELAS** (parte 2):

1. **Um pouco de teoria...**

* Quando criamos uma tabela usando SQL, escolhemos uma coisa chamada ENGINE;
* ENGINE = máquina que vai poder criar registros;



* Ou seja, InnoDB = ENGINE.

A principal característica da ENGINE Inno DB é suportar **chaves estrangeiras** (FK)!

Visto que ela suportam as principais regras de uma boa TRANSAÇÃO.

Por padrão, o MySQL possui dentro dele algums **comandos padrões**, como por exemplo:





**Resumo:**

Ou seja, qualquer solicitação que possa ser feita a um Banco de Dados, e ele vai te atender da melhor maneira possível seguindo os quatro princípios que chamamos de **DICA**.

**D**urabilidade

**I**solamento

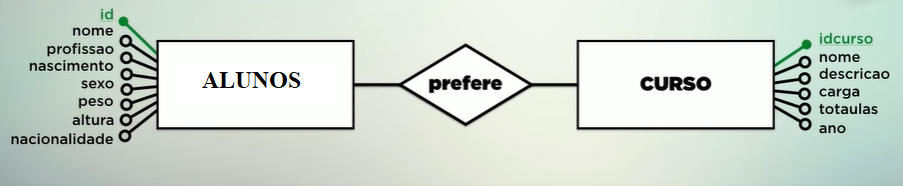
**C**onsistência

**A**tomicidade

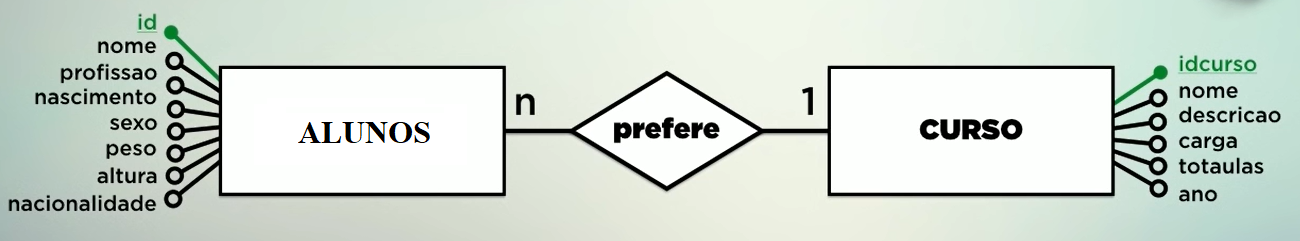
Um breve resumo sobre cada um deles...

1. Durabilidade: Todo dado que é colocado, alterado, ou manipulado tem que permanecer durável. Ou seja, ele tem que permancer dessa maneira, enquanto eu quiser que esse dado esteja lá.
2. Isolamento: Se eu tenho duas transações feitas ao mesmo tempo, elas devem ser executadas sem uma uma interferir na outra. Ou seja, elas são isoladas.
3. Consistência: Toda a transação tem que levar o Banco de Dados de um estado consistente à outro consistente. Ou seja, se tudo estava ‘’OK’’ antes, tudo deve continuar ‘’OK’’ depois.
4. Atomicidade: Toda transação tem que ser atômica Ou seja, ou TODA elas acontece, ou NADA acontece. Ou TUDO dá certo, ou ele dá um ‘’ctrl+z’’ para um estado anteriormente consistente.
5. **Um pouco de prática...**

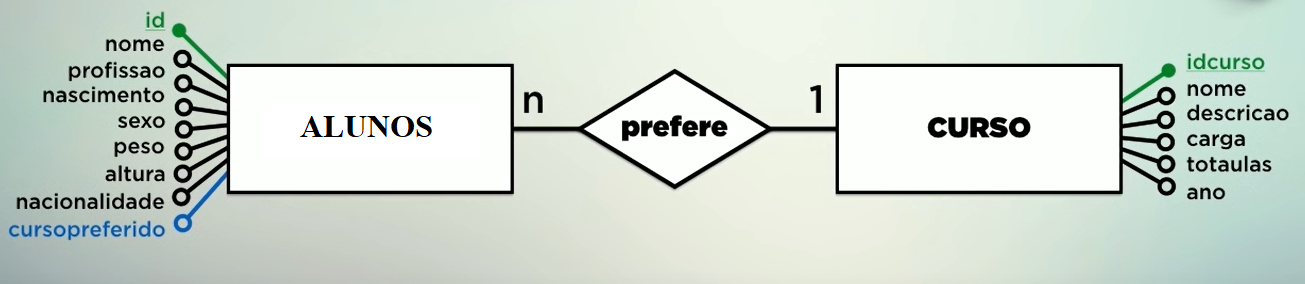
**[Exercício 1]** Baseado nesse RELACIONAMENTO entre as ENTIDADES, qual a cardinalidade dessa relação?



**Resposta**: (1,n)



* Cardinalidade definida, precisamos passar a PK para o lano que há   
  “n” relacional.



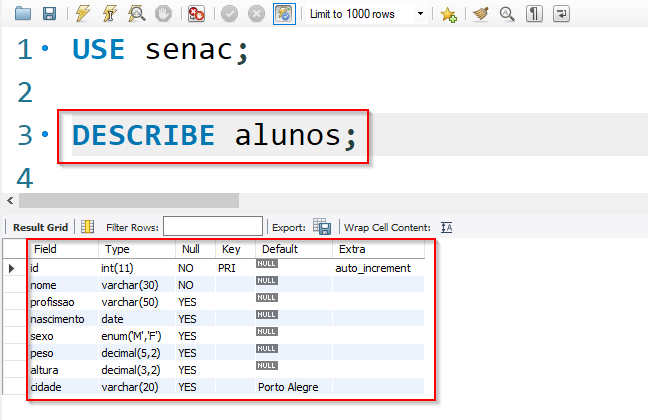
* **LEMBRADO**: A FK não precisa ter o mesmo nome que a PK de onde ela veio. Ela só precisa ter o mesmo TIPO PRIMITIVOS e tamanho.

**3) ADICIONANDO A FOREIGN KEY:**

a) Primeiro passo:

USE senac;

DESCRIBE alunos;

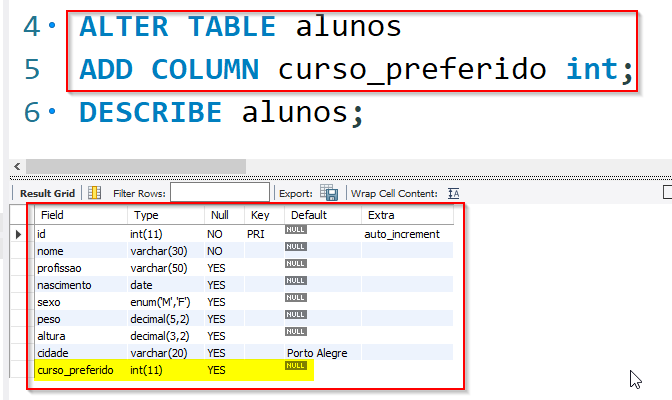


b) Segundo passo:

ALTER TABLE alunos

ADD COLUMN curso\_preferido int;

DESCRIBE alunos;



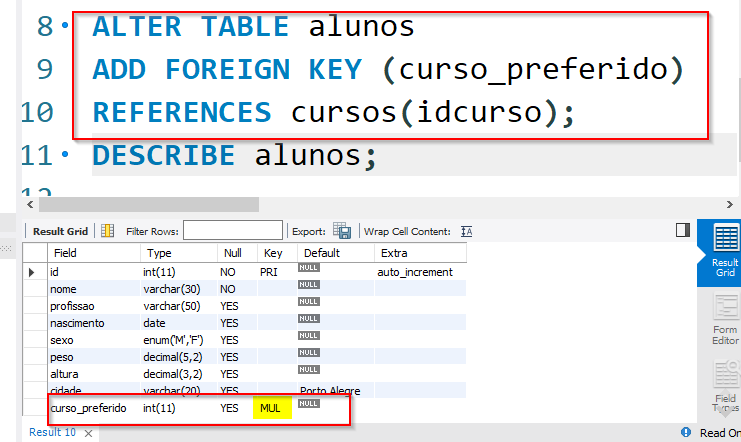
c) Terceiro passo:

\* Precisamos informar que esse novo campo recem adicionado é agora a nossa chave estrangeira (FK).

ALTER TABLE alunos

ADD FOREIGN KEY (curso\_preferido)

REFERENCES cursos(idcurso);



* MUL – É a representação de uma chave multipla, que, no nosso caso é uma chave estrangeira (FK).

**Prática**

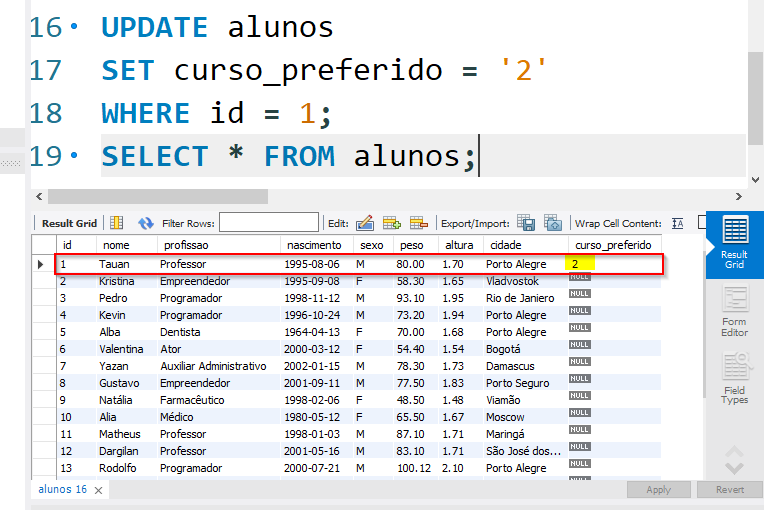
**[Exercício 2]** Visto que agora na tabela “alunos” há um novo campo chamado “curso\_preferido”, você deve adicionar um curso, da tabela “cursos”, para cada um dos resgistros em sua tabela de “alunos”, como o exemplo a seguir.

**Resposta:**

UPDATE alunos

SET curso\_preferido = '2'

WHERE id = 1;



**Ou...**

Esse exercício também pode ser resolvido da maneira direta já ensinada anteriormente.

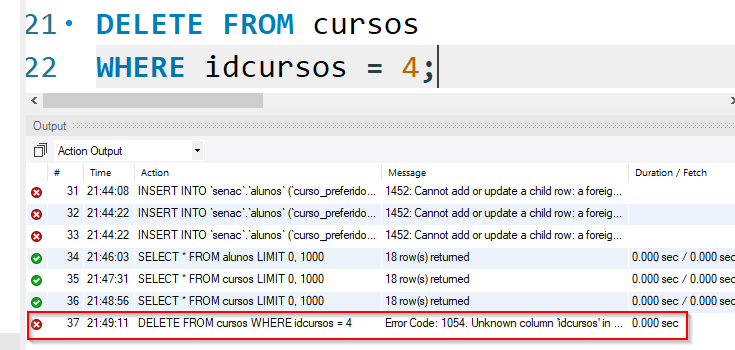
**4) INTEGRADIDA REFERENCIAL:**

**[Exercício 3]** Tente deletar um curso da tabela “cursos” que seja o preferido de pelo menos um aluno.

**Resposta:**

DELETE FROM cursos

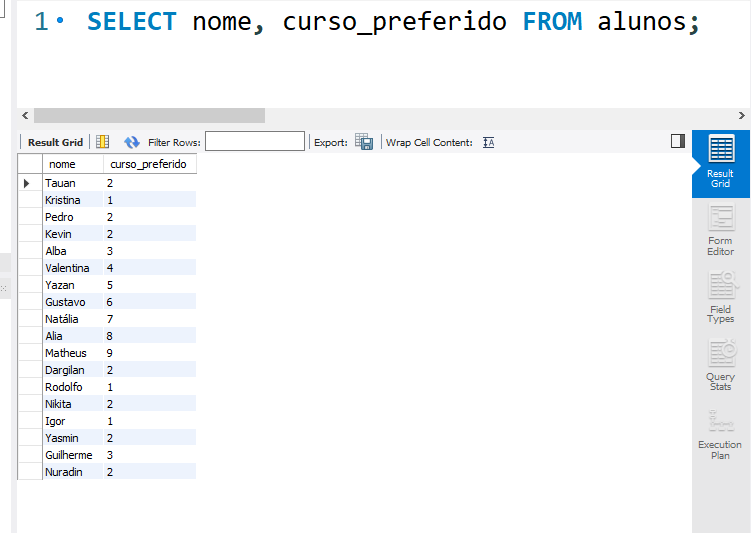
WHERE idcursos = 4;



**[Exercício 4]** Por quê não foi possível deletar o curso?

1. **ENTENDENDO AS RELAÇÕES:**
2. Primeiro passo:

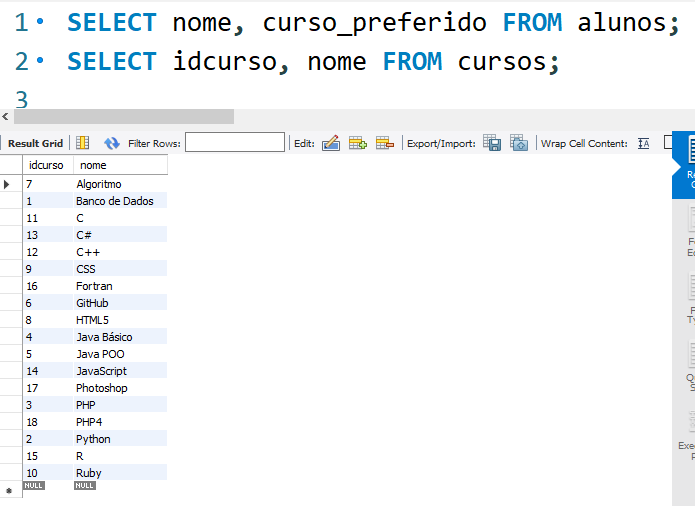
SELECT nome, curso\_preferido FROM alunos;



\* Porém, nós queremos ver o nome do “Curso Preferido” e não o número dele.

SELECT nome, curso\_preferido FROM alunos;

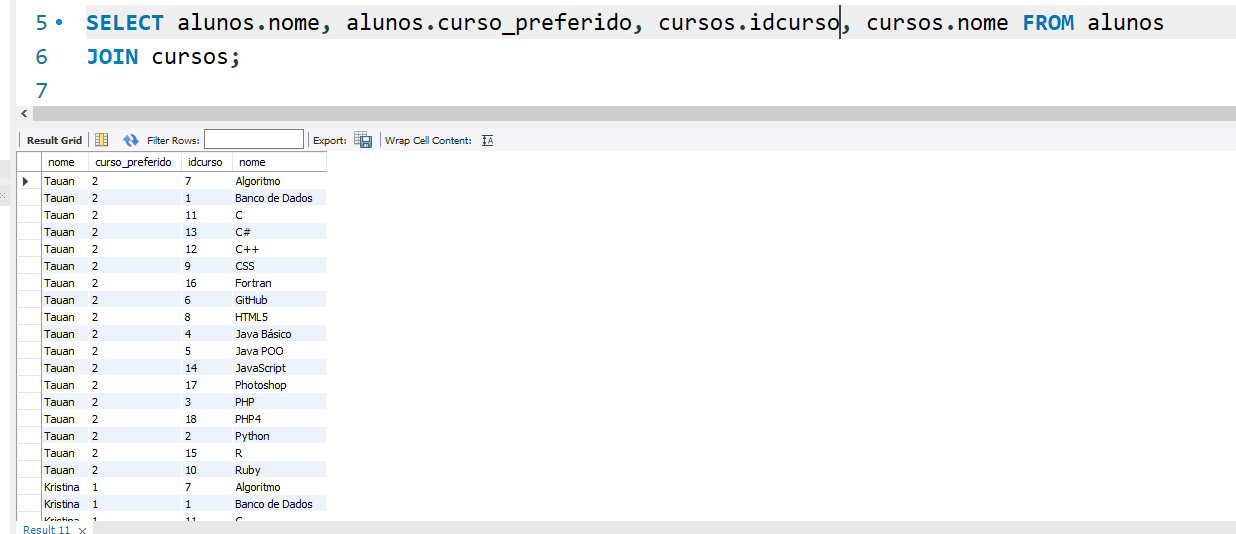
SELECT idcurso, nome FROM cursos;



\* Como juntar essas duas tabelas? A Cláusula “**JOIN**” pode ser a resposta!

1. **O Comando “JOIN”**
2. Primeiro passo:

SELECT alunos.nome, alunos.curso\_preferido, cursos.idcurso, cursos.nome FROM alunos

JOIN cursos; 

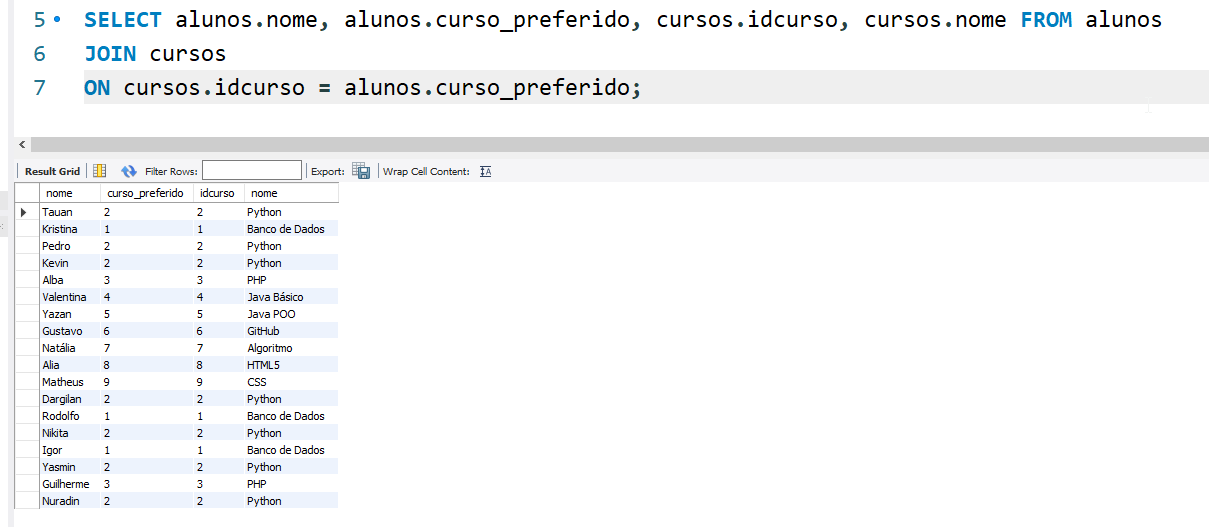
* Como corrigir essa barbária? Cláusula “**ON**”.

1. Segundo passo:

SELECT alunos.nome, alunos.curso\_preferido, cursos.idcurso, cursos.nome FROM alunos

JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido;



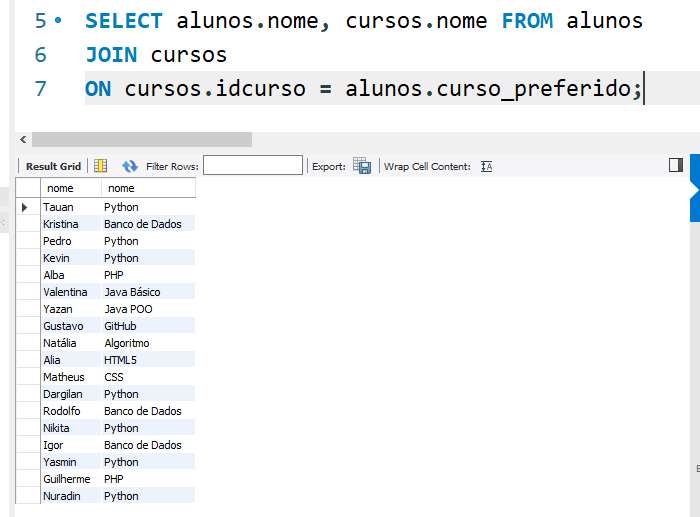
\* Ainda podemos deixar mais claro essa relação.

1. Terceiro passo:

SELECT alunos.nome, cursos.nome FROM alunos

JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido;

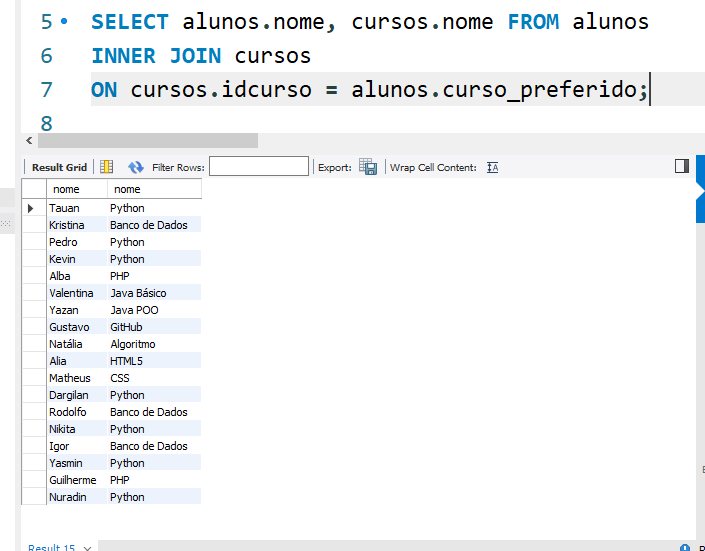


* “By Default”, escrevendo-se somente JOIN, subentende-se que é um **INNER** JOIN. Ou seja, aparecerá APENAS aqueles registros que tem uma relação com algum registro da outra tabela. Aqueles que não tem, ficarão de fora.
* Sendo assim, o comando anterior também poderia ser escrito da seguinte forma:

SELECT alunos.nome, cursos.nome FROM alunos

INNER JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido;



1. Quarto passo:

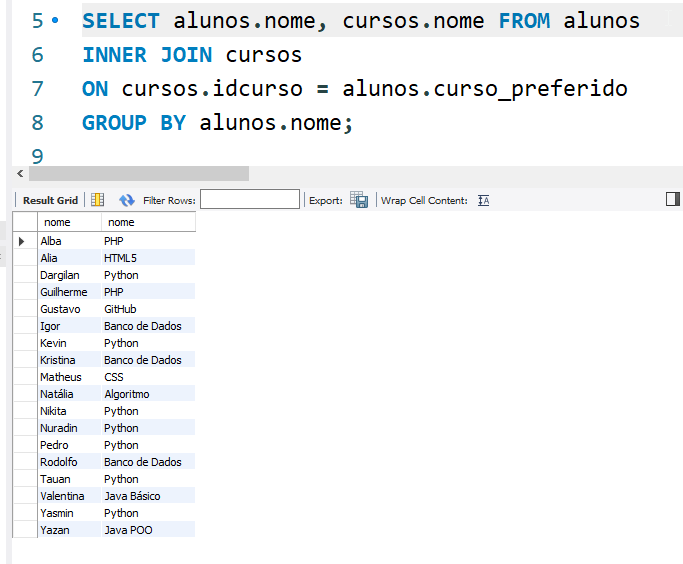
* Podemos também usar outras cláusulas já estudas anteriormente, como, por exemplo, o “GROUP BY”;

SELECT alunos.nome, cursos.nome FROM alunos

INNER JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido

GROUP BY alunos.nome;



1. Quinto passo:

* AGRUPRANDO as colunas ORDENADAS por nome.

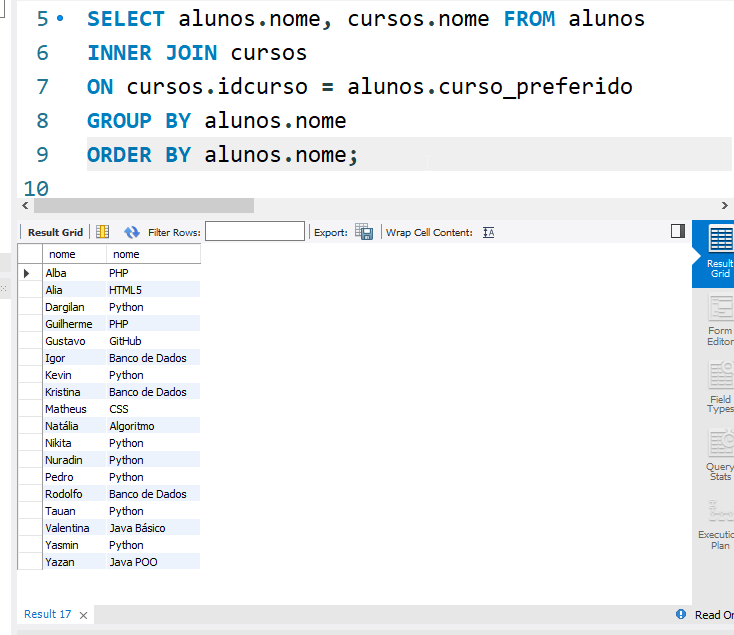
SELECT alunos.nome, cursos.nome FROM alunos

INNER JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido

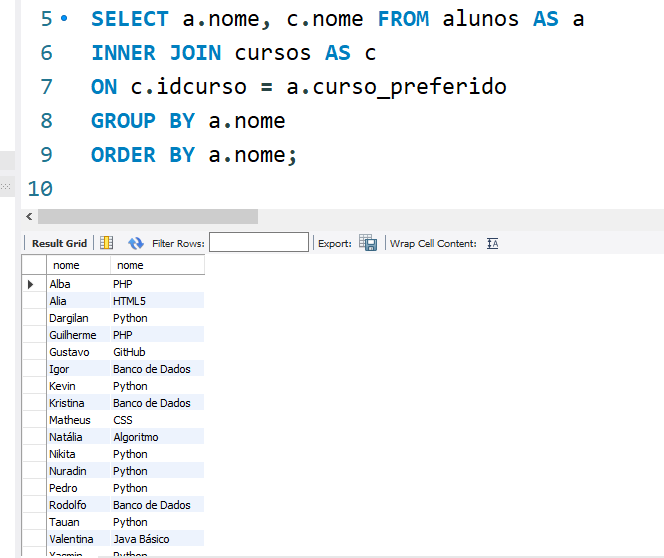
GROUP BY alunos.nome

ORDER BY alunos.nome;



1. Sexto passo:

* Gerando um apelido para as colunas;



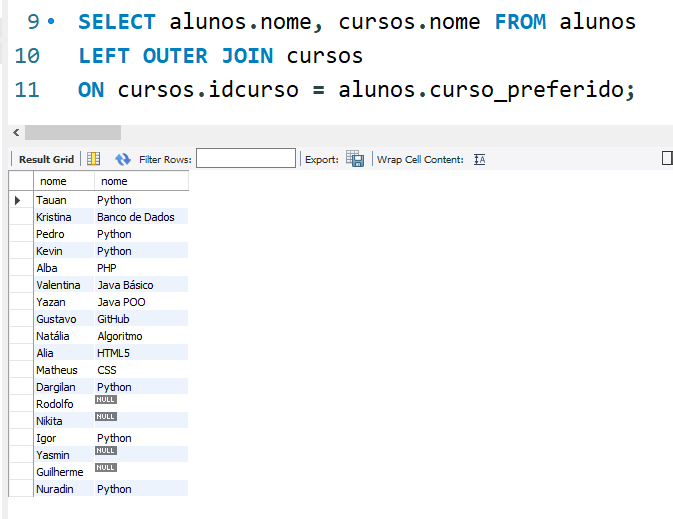
1. Sétimo passo:

* “**OUTER JOIN**” a esquerda (**LEFT OUTER JOIN**) e a direita (**RIGHT OUTER JOIN**);

SELECT alunos.nome, cursos.nome FROM alunos

LEFT OUTER JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido;



SELECT alunos.nome, cursos.nome FROM alunos

RIGHT OUTER JOIN cursos

ON cursos.idcurso = alunos.curso\_preferido;

