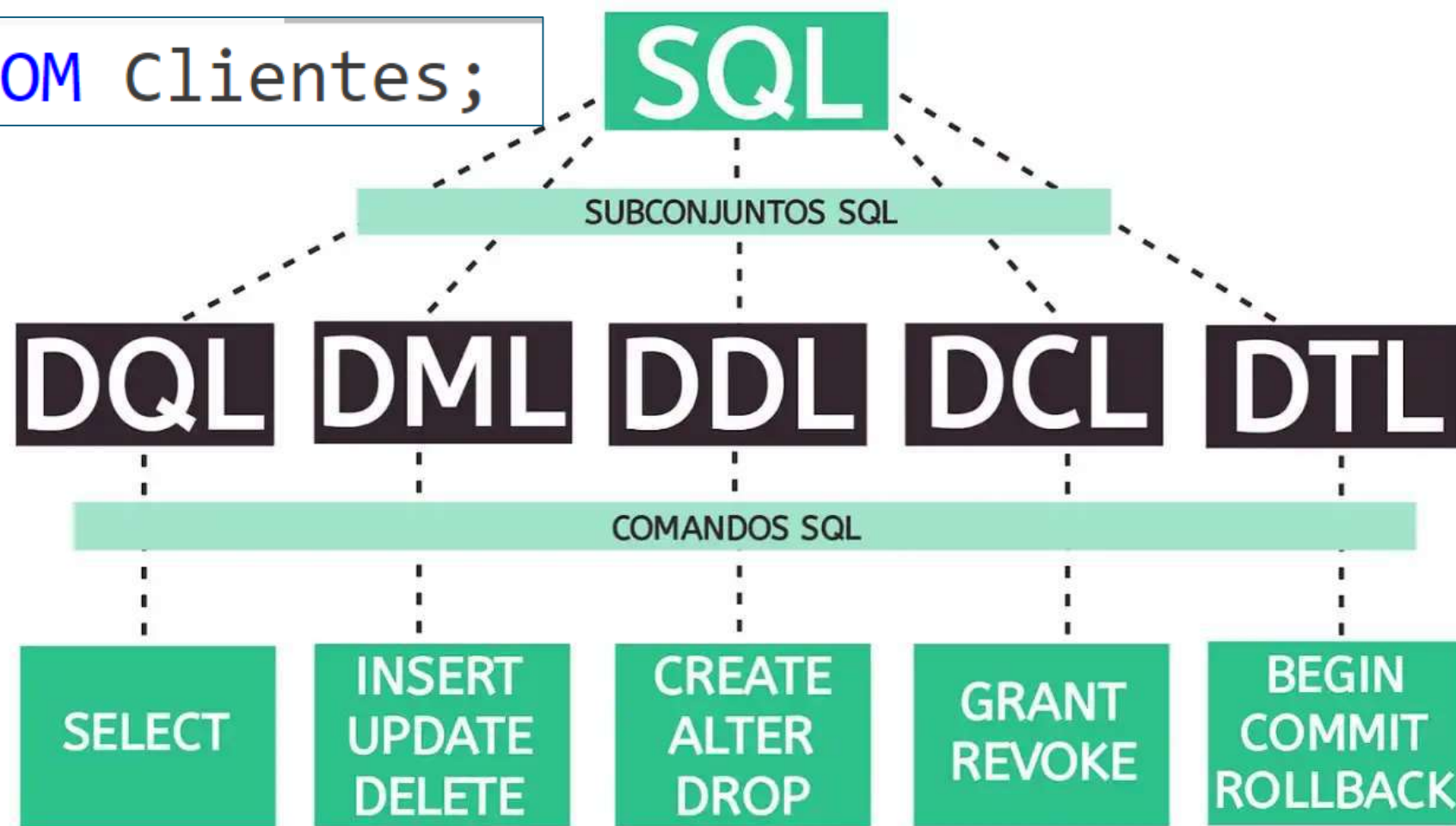






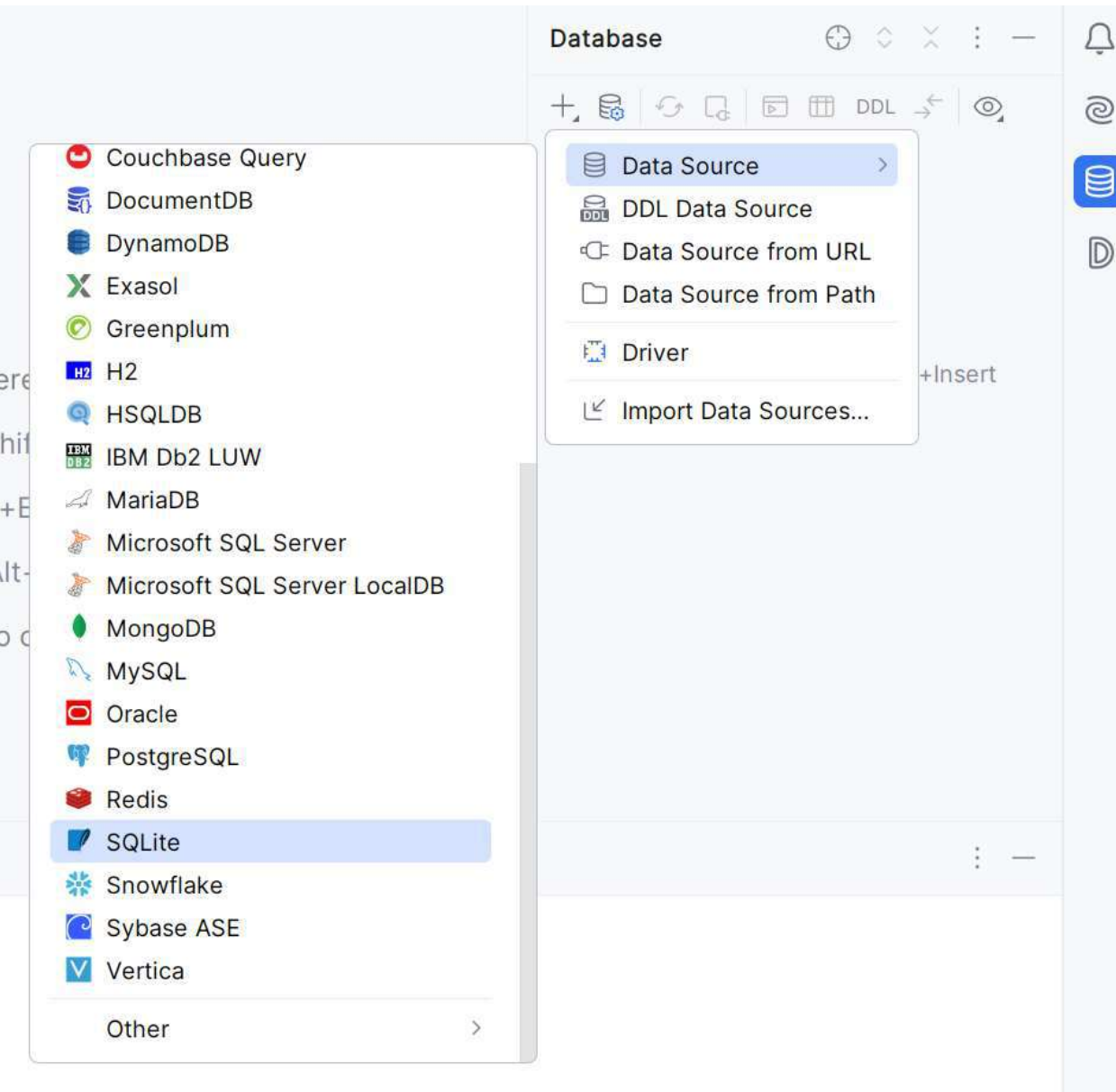
A Linguagem de consulta estruturada (SQL) é uma linguagem padrão para criação e manipulação de bancos de dados.

```
1 SELECT * FROM Clientes;
```

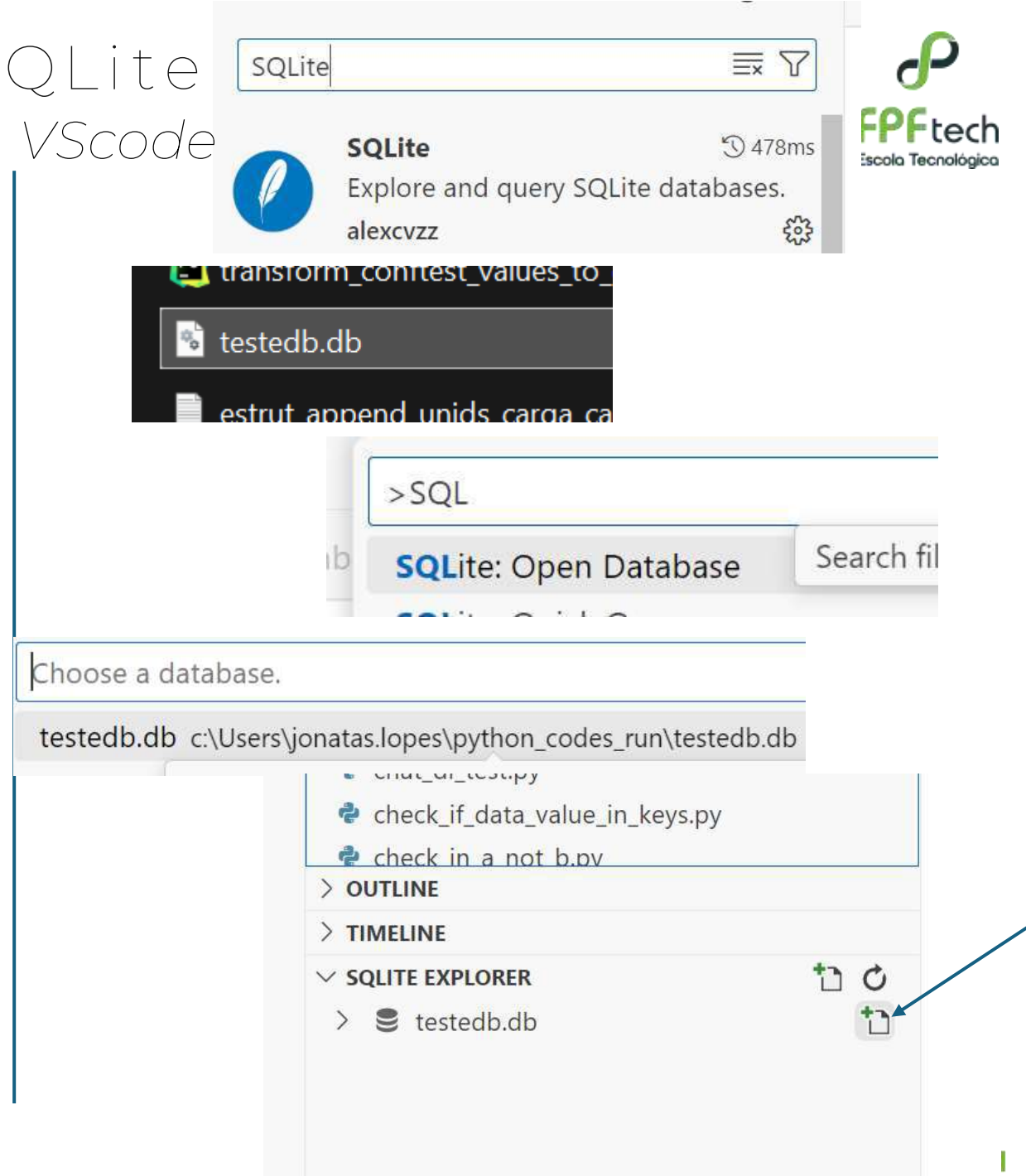


Conectar banco de dados SQLite

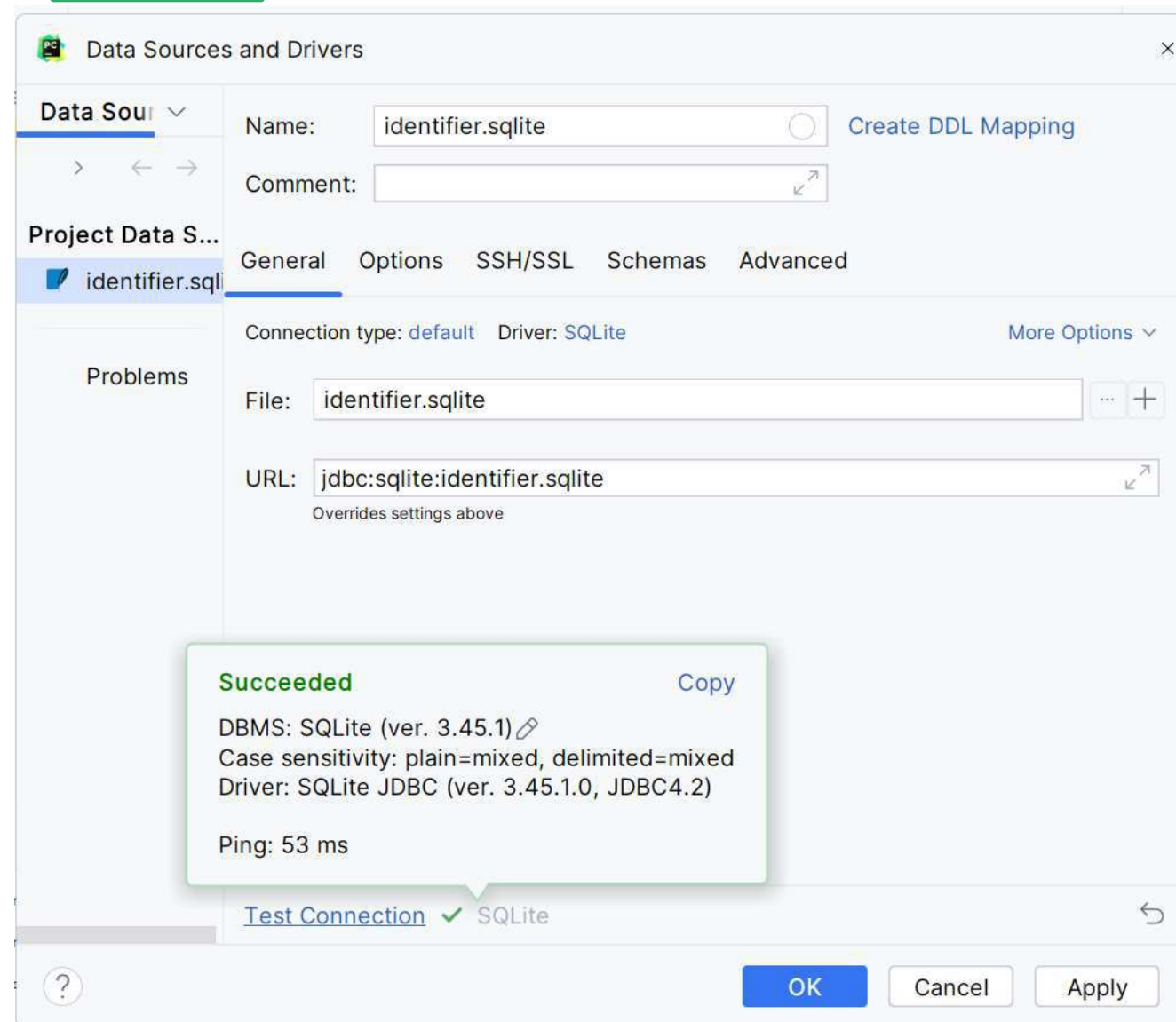
Pycharm



Vscode



Conectar banco de dados SQLite

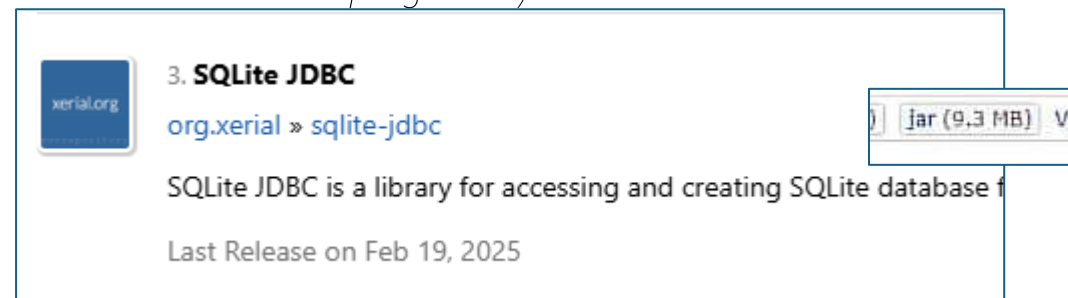


Senão... Precisa instalar:
1 – sqlite download

Precompiled Binaries for Windows

sqlite-dll-win-x86-3490100.zip (1.02 MiB)	32-bit DLL (x86) for SQLite version 3.49.1. (SHA3-256: 1dac243f154828327d6a250b74cf115e4f6a4cbfe83dbc488d45510b59e7a157)
sqlite-dll-win-x64-3490100.zip (1.28 MiB)	64-bit DLL (x64) for SQLite version 3.49.1. (SHA3-256: ec8fb7976d9c4bc495c4a142da05e97c4d6dc6c1205877adce0bd5b191026e2)
sqlite-tools-win-x64-3490100.zip (6.12 MiB)	A bundle of command-line tools for managing SQLite database files, including (SHA3-256: f80860a651026229c8d6ba76cd8c3af74e114cecf71ad9fc5bf938adcd4ca0b9)

2 – Maven repository > SQLite JDBC
(baixar o arq. “jar”)



2.1. – colocar tudo acima numa pasta e colocar a pasta no path, (talvez precise reiniciar)

3 – Testar no Terminal “sqlite3”


```
CREATE TABLE Alunos (  
  ID INT PRIMARY KEY, <- Autoincrement  
  Nome VARCHAR(100),  
  Idade INT,  
  Email VARCHAR(100)  
);
```

Não precisa
mais passar
o id aqui

->

```
INSERT INTO Alunos (ID, Nome, Idade, Email)  
VALUES  
(1, 'João Silva', 20, 'joao.silva@email.com'),  
(2, 'Maria Oliveira', 22, 'maria.oliveira@email.com'),  
(3, 'Carlos Souza', 19, 'carlos.souza@email.com');
```



```
UPDATE Alunos  
SET Nome = 'João Pedro Silva', Idade = 21  
WHERE ID = 1;
```

```
DELETE FROM Alunos  
WHERE ID = 1;
```

```
ALTER TABLE Alunos  
ADD COLUMN Genero VARCHAR(10);
```



```
UPDATE Alunos  
SET Genero = 'Feminino'  
WHERE ID = 2;
```

```
UPDATE Alunos  
SET Genero = 'Masculino'  
WHERE ID = 3;
```

Ou... ->
WHERE ID IN [2,5,6...]

```
SELECT * FROM Alunos;
```

```
SELECT DISTINCT nome FROM Alunos;
```

```
SELECT * FROM Alunos ORDER BY nome;
```

```
SELECT * FROM Alunos WHERE idade=20;
```

```
SELECT * FROM Alunos WHERE idade > 20;
```



```
SELECT * FROM Alunos WHERE Idade > 20 AND Nome = 'Maria Oliveira';
```

```
SELECT * FROM Alunos WHERE Idade > 20 OR Nome = 'Maria Oliveira';
```

```
SELECT * FROM Alunos WHERE NOT Idade = 20;
```

```
SELECT * FROM Alunos WHERE (Idade > 20 AND Nome = 'Maria Oliveira') OR Idade = 19;
```




```
SELECT COUNT(*) FROM Alunos;
```

```
SELECT SUM(idade) FROM Alunos;
```

```
SELECT AVG(idade) FROM Alunos;
```

```
SELECT MAX(idade) FROM Alunos;
```

```
SELECT MIN(idade) FROM Alunos;
```



Employee ID	Name	Department
1001	Kundan	Sales
1002	Virat	Marketing
1003	Santosh	Education
1004	Veer	Marketing
1005	Shivani	Sales
1006	Yogesh	Education

GROUP BY Department

Department
Sales
Marketing
Education

SELECT COUNT(*) FROM Alunos GROUP BY Genero

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1

genre	total
adventure	7
fantasy	8
romance	3

```
CREATE TABLE Materias (  
  ID INT PRIMARY KEY,  
  Aluno_ID INT,  
  Nome VARCHAR(50),  
  FOREIGN KEY (Aluno_ID) REFERENCES Alunos(ID)  
);
```

JustPaste.it

INSERT



@anonymous - 8/08/2024

```
INSERT INTO Alunos (ID, Nome, Idade, Email, Genero) VALUES  
(1, 'Ana Silva', 20, 'ana.silva@example.com', 'Feminino'),  
(2, 'Bruno Costa', 22, 'bruno.costa@example.com', 'Masculino'),  
(3, 'Carla Oliveira', 21, 'carla.oliveira@example.com', 'Feminino'),  
(4, 'Daniel Santos', 23, 'daniel.santos@example.com', 'Masculino'),  
(5, 'Eduarda Lima', 19, 'eduarda.lima@example.com', 'Feminino'),  
(6, 'Felipe Almeida', 24, 'felipe.almeida@example.com', 'Masculino'),  
(7, 'Gabriela Pereira', 22, 'gabriela.pereira@example.com', 'Feminino'),  
(8, 'Henrique Sousa', 20, 'henrique.sousa@example.com', 'Masculino'),  
(9, 'Isabela Rodrigues', 21, 'isabela.rodrigues@example.com', 'Feminino'),  
(10, 'João Lima', 25, 'joao.lima@example.com', 'Masculino').
```

<https://justpaste.it/hlija>

Rodar o INSERT de Alunos, depois o de Materias



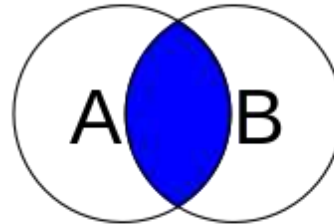
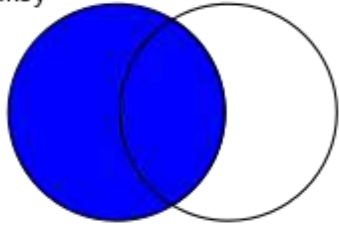
@fpftech.educacional

Joins (junção de tabelas)

```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

Left

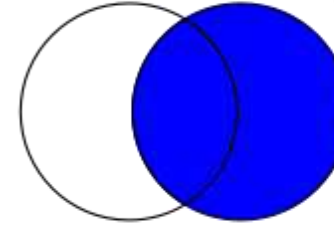
```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```



inner

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

right



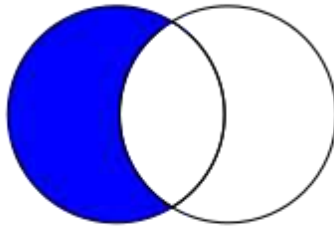
SQL

JOINS

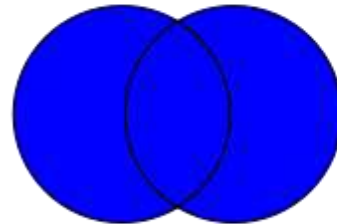
Left

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

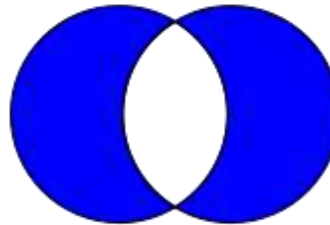
where



outer



Full outer



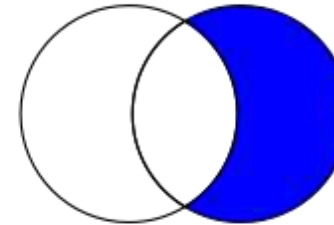
where

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL
```

right

where



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```



Joins (junção de tabelas)

```
SELECT * FROM Alunos INNER JOIN Materias M on Alunos.ID = M.Aluno_ID
```

Apenas onde há uma correspondência em ambas as tabelas envolvidas.

```
SELECT * FROM Alunos LEFT JOIN Materias M on Alunos.ID = M.Aluno_ID
```

As linhas da tabela à esquerda (a tabela principal) e as linhas correspondentes da tabela à direita (a tabela secundária). Se não houver correspondência, o resultado ainda inclui todas as linhas da tabela à esquerda, mas com valores NULL para as colunas da tabela à direita.



<https://mystery.knightlab.com/>



Obrigado!



    @fpftech.educacional