

# Questionário Médio de SQL

---

## Banco de Dados: Biblioteca

---

### Estrutura:

1. **Livro** (ID\_Livro, Titulo, Autor, Ano\_Publicacao, ID\_Editora, Disponivel)
2. **Editora** (ID\_Editora, Nome, Cidade)
3. **Usuario** (ID\_Usuario, Nome, Email, Data\_Cadastro, Tipo)
4. **Emprestimo** (ID\_Emprestimo, ID\_Livro, ID\_Usuario, Data\_Emprestimo, Data\_Devolucao)
5. **Multa** (ID\_Multa, ID\_Emprestimo, Valor, Data\_Pagamento)

## Parte A: Construção do Banco (5 questões)

---

### A1. Crie a tabela `Editora` com:

- `ID_Editora` (int, PK, auto incremento)
- `Nome` (varchar(100), não nulo, único)
- `Cidade` (varchar(50), não nulo)

### A2. Crie a tabela `Livro` com:

- `ID_Livro` (int, PK, auto incremento)
- `Titulo` (varchar(200), não nulo)
- `Autor` (varchar(100), não nulo)
- `Ano_Publicacao` (int, entre 1500 e 2025)
- `ID_Editora` (int, FK para Editora, não nulo)
- `Disponivel` (bit, padrão 1)

**A3.** Crie a tabela `Usuario` com:

- `ID_Usuario` (int, PK, auto incremento)
- `Nome` (varchar(100), não nulo)
- `Email` (varchar(100), único)
- `Data_Cadastro` (date, padrão data atual)
- `Tipo` (varchar(20), deve ser 'Estudante', 'Professor' ou 'Comunidade')

**A4.** Crie a tabela `Emprestimo` com:

- `ID_Emprestimo` (int, PK, auto incremento)
- `ID_Livro` (int, FK para Livro, não nulo)
- `ID_Usuario` (int, FK para Usuario, não nulo)
- `Data_Emprestimo` (date, não nulo, padrão data atual)
- `Data_Devolucao` (date, deve ser maior ou igual a `Data_Emprestimo`)

**A5.** Crie a tabela `Multa` com:

- `ID_Multa` (int, PK, auto incremento)
- `ID_Emprestimo` (int, FK para Emprestimo, não nulo, único)
- `Valor` (decimal(10,2), não nulo, maior que 0)
- `Data_Pagamento` (date, pode ser nulo)

## Parte B: Consultas (15 questões)

---

1. Liste todos os livros disponíveis, mostrando título, autor e nome da editora.
2. Conte quantos livros cada editora possui, ordenando da editora com mais livros para a com menos.
3. Liste os empréstimos atuais (não devolvidos) com nome do usuário e título do livro.
4. Calcule o número de dias de atraso para cada empréstimo em atraso (`Data_Devolucao < hoje` e ainda não devolvido).
5. Insira uma nova editora e dois livros dessa editora.

6. Atualize o status de disponibilidade para falso quando um livro for emprestado.
  7. Crie uma consulta que mostre o número total de empréstimos por usuário no último mês.
  8. Liste os usuários que têm multas não pagas (Data\_Pagamento nulo e Valor > 0).
  9. Crie uma view que mostre os livros mais populares (com mais empréstimos).
  10. Crie um índice na coluna Autor da tabela Livro.
  11. Escreva uma consulta que use LIKE para buscar livros com a palavra “banco” no título.
  12. Use GROUP BY e HAVING para listar autores que têm mais de 3 livros no acervo.
  13. Utilize INNER JOIN para listar todos os empréstimos com detalhes do livro, usuário e editora.
  14. Escreva uma subquery para encontrar usuários que nunca emprestaram livros.
  15. Crie uma stored procedure que registre a devolução de um livro, atualizando a data de devolução e a disponibilidade do livro.
- 

## Respostas - Questionário Médio

---

### Parte A

A1:

```
CREATE TABLE Editora (  
    ID_Editora INT PRIMARY KEY IDENTITY(1,1),  
    Nome VARCHAR(100) NOT NULL UNIQUE,  
    Cidade VARCHAR(50) NOT NULL  
);
```

A2:

```
CREATE TABLE Livro (
    ID_Livro INT PRIMARY KEY IDENTITY(1,1),
    Titulo VARCHAR(200) NOT NULL,
    Autor VARCHAR(100) NOT NULL,
    Ano_Publicacao INT NOT NULL,
    ID_Editora INT NOT NULL,
    Disponivel BIT DEFAULT 1,
    CONSTRAINT CHK_Ano_Publicacao CHECK (Ano_Publicacao BETWEEN 1500 AND 2025),
    CONSTRAINT FK_Livro_Editora FOREIGN KEY (ID_Editora) REFERENCES
Editora(ID_Editora)
);
```

**A3:**

```
CREATE TABLE Usuario (
    ID_Usuario INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Data_Cadastro DATE DEFAULT GETDATE(),
    Tipo VARCHAR(20) NOT NULL,
    CONSTRAINT CHK_Tipo CHECK (Tipo IN ('Estudante', 'Professor',
'Comunidade'))
);
```

**A4:**

```
CREATE TABLE Emprestimo (
    ID_Emprestimo INT PRIMARY KEY IDENTITY(1,1),
    ID_Livro INT NOT NULL,
    ID_Usuario INT NOT NULL,
    Data_Emprestimo DATE NOT NULL DEFAULT GETDATE(),
    Data_Devolucao DATE,
    CONSTRAINT FK_Emprestimo_Livro FOREIGN KEY (ID_Livro) REFERENCES
Livro(ID_Livro),
    CONSTRAINT FK_Emprestimo_Usuario FOREIGN KEY (ID_Usuario) REFERENCES
Usuario(ID_Usuario),
    CONSTRAINT CHK_Data_Devolucao CHECK (Data_Devolucao >= Data_Emprestimo)
);
```

A5:

```
CREATE TABLE Multa (  
    ID_Multa INT PRIMARY KEY IDENTITY(1,1),  
    ID_Emprestimo INT NOT NULL UNIQUE,  
    Valor DECIMAL(10,2) NOT NULL,  
    Data_Pagamento DATE,  
    CONSTRAINT FK_Multa_Emprestimo FOREIGN KEY (ID_Emprestimo) REFERENCES  
    Emprestimo(ID_Emprestimo),  
    CONSTRAINT CHK_Valores CHECK (Valor > 0)  
);
```

## Parte B

1:

```
SELECT L.Titulo, L.Autor, E.Nome AS Editora  
FROM Livro L  
INNER JOIN Editora E ON L.ID_Editora = E.ID_Editora  
WHERE L.Disponivel = 1;
```

2:

```
SELECT E.Nome, COUNT(L.ID_Livro) AS Total_Livros  
FROM Editora E  
LEFT JOIN Livro L ON E.ID_Editora = L.ID_Editora  
GROUP BY E.Nome  
ORDER BY Total_Livros DESC;
```

3:

```
SELECT U.Nome, L.Titulo  
FROM Emprestimo EM  
INNER JOIN Usuario U ON EM.ID_Usuario = U.ID_Usuario  
INNER JOIN Livro L ON EM.ID_Livro = L.ID_Livro  
WHERE EM.Data_Devolucao IS NULL;
```

4:

```
SELECT ID_Emprestimo, DATEDIFF(DAY, Data_Emprestimo, GETDATE()) AS
Dias_Atraso
FROM Emprestimo
WHERE Data_Devolucao IS NULL AND GETDATE() > DATEADD(DAY, 30,
Data_Emprestimo);
-- Considerando que o prazo é 30 dias
```

5:

```
INSERT INTO Editora (Nome, Cidade) VALUES ('Nova Editora', 'Rio de
Janeiro');
DECLARE @EditoraID INT = SCOPE_IDENTITY();

INSERT INTO Livro (Titulo, Autor, Ano_Publicacao, ID_Editora) VALUES
('Livro A', 'Autor A', 2020, @EditoraID),
('Livro B', 'Autor B', 2021, @EditoraID);
```

6: (Assumindo que quando um empréstimo é inserido, o livro fica indisponível)

```
-- Inserir empréstimo e atualizar livro
BEGIN TRANSACTION;
INSERT INTO Emprestimo (ID_Livro, ID_Usuario) VALUES (1, 1);
UPDATE Livro SET Disponivel = 0 WHERE ID_Livro = 1;
COMMIT;
```

7:

```
SELECT U.Nome, COUNT(EM.ID_Emprestimo) AS Total_Emprestimos
FROM Usuario U
LEFT JOIN Emprestimo EM ON U.ID_Usuario = EM.ID_Usuario
WHERE EM.Data_Emprestimo >= DATEADD(MONTH, -1, GETDATE())
GROUP BY U.Nome;
```

8:

```
SELECT U.Nome, M.Valor
FROM Multa M
INNER JOIN Emprestimo EM ON M.ID_Emprestimo = EM.ID_Emprestimo
INNER JOIN Usuario U ON EM.ID_Usuario = U.ID_Usuario
WHERE M.Data_Pagamento IS NULL AND M.Valor > 0;
```

9:

```
CREATE VIEW Livros_Populares AS
SELECT L.Titulo, COUNT(EM.ID_Emprestimo) AS Num_Emprestimos
FROM Livro L
LEFT JOIN Emprestimo EM ON L.ID_Livro = EM.ID_Livro
GROUP BY L.Titulo;
```

10:

```
CREATE INDEX IX_Livro_Autor ON Livro(Autor);
```

11:

```
SELECT * FROM Livro WHERE Titulo LIKE '%banco%';
```

12:

```
SELECT Autor, COUNT(*) AS Total_Livros
FROM Livro
GROUP BY Autor
HAVING COUNT(*) > 3;
```

13:

```
SELECT EM.Data_Emprestimo, L.Titulo, U.Nome AS Usuario, E.Nome AS Editora
FROM Emprestimo EM
INNER JOIN Livro L ON EM.ID_Livro = L.ID_Livro
INNER JOIN Usuario U ON EM.ID_Usuario = U.ID_Usuario
INNER JOIN Editora E ON L.ID_Editora = E.ID_Editora;
```

14:

```
SELECT * FROM Usuario
WHERE ID_Usuario NOT IN (SELECT DISTINCT ID_Usuario FROM Emprestimo);
```

15:

```
CREATE PROCEDURE sp_DevolverLivro
    @ID_Emprestimo INT
AS
BEGIN
    BEGIN TRANSACTION;
    UPDATE Emprestimo SET Data_Devolucao = GETDATE() WHERE ID_Emprestimo =
@ID_Emprestimo;
    UPDATE Livro SET Disponivel = 1 WHERE ID_Livro = (SELECT ID_Livro FROM
Emprestimo WHERE ID_Emprestimo = @ID_Emprestimo);
    COMMIT;
END;
```