# TODOS OS SCRIPTS SQL USADOS PARA CRIAR O BANCO DE DADOS DO SISTEMA

```
-- Banco de Dados: bd_biblioteca_online
CREATE DATABASE bd_biblioteca_online
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'pt-BR'
    LC_CTYPE = 'pt-BR'
    LOCALE_PROVIDER = 'libc'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;
COMMENT ON DATABASE bd_biblioteca_online
    IS 'Banco de dados do projeto Sistema de Biblioteca Online';

-- Tabela de Livros.
CREATE TABLE Livros (
 id SERIAL PRIMARY KEY,
 titulo VARCHAR(255) NOT NULL,
 autor VARCHAR(255) NOT NULL,
 isbn VARCHAR(20) UNIQUE,
 ano_publicacao INTEGER,
 editora VARCHAR(100),
 quantidade_total INTEGER NOT NULL,
 quantidade_disponivel INTEGER NOT NULL,
 descricao TEXT,
 numero_paginas INTEGER,
 genero VARCHAR(100));

 -- Tabela de Usuários.
CREATE TABLE Usuarios (
 id SERIAL PRIMARY KEY,
 nome VARCHAR(255) NOT NULL,
 tipo VARCHAR(50) NOT NULL CHECK (tipo IN ('aluno', 'professor', 'pesquisador',
'administrador')),
 email VARCHAR(255) UNIQUE,
 telefone VARCHAR(20) NOT NULL,
 password   VARCHAR(255) NOT NULL,
 last_login  TIMESTAMP  NULL,
 punido_ate DATE
```

```
);



-- Tabela de Exemplares (Cópias de Livros Físicos).
CREATE TABLE Exemplares (
 id SERIAL PRIMARY KEY,
 id_livro INTEGER REFERENCES Livros(id),
 codigo_barras VARCHAR(100) UNIQUE,
 status VARCHAR(50) CHECK (status IN ('disponivel', 'emprestado', 'reservado',
'indisponivel', 'perdido')) DEFAULT 'disponivel'
);

-- Tabela de Recursos Digitais.
CREATE TABLE RecursosDigitais (
 id SERIAL PRIMARY KEY,
 titulo VARCHAR(255) NOT NULL,
 tipo VARCHAR(50) NOT NULL CHECK (tipo IN ('e-book', 'periodico', 'tese',
'dissertacao', 'base_de_dados')),
 url TEXT NOT NULL,
 data_disponibilidade DATE,
 descricao TEXT,
 numero_paginas INTEGER,
 genero VARCHAR(100)
);

-- Tabela de Reservas.
CREATE TABLE Reservas (
 id SERIAL PRIMARY KEY,
 id_exemplar INTEGER REFERENCES Exemplares(id),
 id_usuario INTEGER REFERENCES Usuarios(id),
 data_reserva TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
 status VARCHAR(20)
CHECK (status IN ('pendente', 'aguardando_confirmacao', 'cancelada', 'finalizada'))
NULL;
);

-- Tabela de Empréstimos.
CREATE TABLE Emprestimos (
 id SERIAL PRIMARY KEY,
 id_exemplar INTEGER REFERENCES Exemplares(id),
 id_usuario INTEGER REFERENCES Usuarios(id),
 data_emprestimo DATE NOT NULL DEFAULT CURRENT_DATE,
 data_devolucao_prevista DATE NOT NULL,
```

```
 data_devolucao_real DATE,
 status VARCHAR(20)
CHECK (status IN ('pendente', 'ativo', 'concluido', 'cancelado'))
NULL;
);

-- Tabela de Listas de Leitura.
CREATE TABLE ListasLeitura (
 id SERIAL PRIMARY KEY,
 nome VARCHAR(255) NOT NULL,
 descricao TEXT,
 id_usuario INTEGER REFERENCES Usuarios(id)
);

-- Tabela de Livros em Listas de Leitura (relação muitos-para-muitos).
CREATE TABLE LivrosListasLeitura (
 id_lista INTEGER REFERENCES ListasLeitura(id),
 id_livro INTEGER REFERENCES Livros(id),
 PRIMARY KEY (id_lista, id_livro)
);

-- Tabela de Recursos Digitais em Listas de Leitura (relação muitos-para-muitos).
CREATE TABLE RecursosDigitaisListasLeitura (
 id_lista INTEGER REFERENCES ListasLeitura(id),
 id_recurso_digital INTEGER REFERENCES RecursosDigitais(id),
 PRIMARY KEY (id_lista, id_recurso_digital)
);

-- Tabela de Histórico de Empréstimos.
CREATE TABLE HistoricoEmprestimos (
 id SERIAL PRIMARY KEY,
 id_exemplar INTEGER REFERENCES Exemplares(id),
 id_usuario INTEGER REFERENCES Usuarios(id),
 data_emprestimo DATE NOT NULL,
 data_devolucao_prevista DATE NOT NULL,
 data_devolucao_real DATE NOT NULL,
 data_registro_historico TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP
);

-- Tabela de Favoritos.
CREATE TABLE Favoritos (
 id SERIAL PRIMARY KEY,
 id_usuario INTEGER REFERENCES Usuarios(id),
```

```
 id_livro INTEGER REFERENCES Livros(id),
 id_recurso_digital INTEGER REFERENCES RecursosDigitais(id),
 data_favoritado TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
 UNIQUE (id_usuario, id_livro, id_recurso_digital),
 CONSTRAINT fk_favoritos_livro FOREIGN KEY (id_livro) REFERENCES Livros(id)
ON DELETE CASCADE,
 CONSTRAINT fk_favoritos_recurso_digital FOREIGN KEY (id_recurso_digital)
REFERENCES RecursosDigitais(id) ON DELETE CASCADE,
 CONSTRAINT fk_favoritos_usuario FOREIGN KEY (id_usuario) REFERENCES
Usuarios(id) ON DELETE CASCADE,
15
 CONSTRAINT check_favorito_item CHECK (id_livro IS NOT NULL OR
id_recurso_digital IS NOT NULL),
 CONSTRAINT check_favorito_exclusivo CHECK (NOT (id_livro IS NOT NULL AND
id_recurso_digital IS NOT NULL))
);

-- Tabela de Notificações
CREATE TABLE notificacoes (
    id SERIAL PRIMARY KEY,
    usuario_id INTEGER NOT NULL REFERENCES usuarios(id) ON DELETE
CASCADE,
    mensagem TEXT NOT NULL,
    lida BOOLEAN DEFAULT FALSE,
    data_criacao TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP
);

-- Inserindo alguns livros.
INSERT INTO Livros (titulo, autor, isbn, ano_publicacao, editora, quantidade_total,
quantidade_disponivel, descricao, numero_paginas, genero) VALUES
('Dom Casmurro', 'Machado de Assis', '9788508125459', 1899, 'Ática', 5, 5, 'Um
clássico da literatura brasileira que narra a história de Bentinho e Capitu, explorando
temas como ciúme e traição.', 320, 'Romance'),
('O Pequeno Príncipe', 'Antoine de Saint-Exupéry', '9788595080313', 1943,
'Gutenberg', 10, 10, 'Uma fábula poética e filosófica sobre a amizade, o amor e a
perda, contada através da perspectiva de um jovem príncipe.', 96, 'Literatura
Infantil'),
('1984', 'George Orwell', '9788532530785', 1949, 'Companhia das Letras', 7, 7, 'Um
romance distópico que descreve uma sociedade totalitária sob vigilância constante
do Grande Irmão.', 328, 'Ficção Científica'),
('Sapiens: Uma Breve História da Humanidade', 'Yuval Noah Harari',
'9788535927536', 2015, 'Companhia das Letras', 12, 12, 'Uma análise abrangente
```

da história da humanidade, desde os primeiros humanos até o presente.', 496, 'Não Ficção'),
('A Metamorfose', 'Franz Kafka', '9788572326758', 1915, 'Penguin-Companhia', 3, 3, 'A história de Gregor Samsa, que um dia acorda transformado em um inseto monstruoso.', 80, 'Ficção'),
('Orgulho e Preconceito', 'Jane Austen', '9788578270976', 1813, 'Martin Claret', 8, 8, 'Um romance que acompanha os relacionamentos amorosos e as questões sociais da Inglaterra do século XIX.', 432, 'Romance'),
('Cem Anos de Solidão', 'Gabriel García Márquez', '9788501076337', 1967, 'Record', 9, 9, 'A saga da família Buendía ao longo de sete gerações na cidade fictícia de Macondo.', 416, 'Ficção'),
('Neuromancer', 'William Gibson', '9788505011273', 1984, 'Aleph', 6, 6, 'Um romance cyberpunk que segue as aventuras de um talentoso hacker contratado para um último trabalho.', 336, 'Ficção Científica');

-- Exemplares para 'Dom Casmurro' (id_livro = 1).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(1, 'DOMC001'), (1, 'DOMC002'), (1, 'DOMC003'), (1, 'DOMC004'), (1, 'DOMC005');
-- Exemplares para 'O Pequeno Príncipe' (id_livro = 2).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(2, 'PEQP001'), (2, 'PEQP002'), (2, 'PEQP003'), (2, 'PEQP004'), (2, 'PEQP005'),
(2, 'PEQP006'), (2, 'PEQP007'), (2, 'PEQP008'), (2, 'PEQP009'), (2, 'PEQP010');
-- Exemplares para '1984' (id_livro = 3).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(3, 'NINE001'), (3, 'NINE002'), (3, 'NINE003'), (3, 'NINE004'), (3, 'NINE005'),
(3, 'NINE006'), (3, 'NINE007');
-- Exemplares para 'Sapiens' (id_livro = 4).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(4, 'SAPI001'), (4, 'SAPI002'), (4, 'SAPI003'), (4, 'SAPI004'), (4, 'SAPI005'),
(4, 'SAPI006'), (4, 'SAPI007'), (4, 'SAPI008'), (4, 'SAPI009'), (4, 'SAPI010'),
(4, 'SAPI011'), (4, 'SAPI012');
-- Exemplares para 'A Metamorfose' (id_livro = 5).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(5, 'META001'), (5, 'META002'), (5, 'META003');
-- Exemplares para 'Orgulho e Preconceito' (id_livro = 6).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(6, 'ORGU001'), (6, 'ORGU002'), (6, 'ORGU003'), (6, 'ORGU004'), (6, 'ORGU005'),
(6, 'ORGU006'), (6, 'ORGU007'), (6, 'ORGU008');
-- Exemplares para 'Cem Anos de Solidão' (id_livro = 7).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES
(7, 'CEMM001'), (7, 'CEMM002'), (7, 'CEMM003'), (7, 'CEMM004'), (7, 'CEMM005'),
(7, 'CEMM006'), (7, 'CEMM007'), (7, 'CEMM008'), (7, 'CEMM009');
-- Exemplares para 'Neuromancer' (id_livro = 8).
INSERT INTO Exemplares (id_livro, codigo_barras) VALUES

(8, 'NEUR001'), (8, 'NEUR002'), (8, 'NEUR003'), (8, 'NEUR004'), (8, 'NEUR005'), (8, 'NEUR006');

- Inserção de alguns usuários.
INSERT INTO Usuarios (nome, tipo, email, telefone) VALUES
('João Silva', 'aluno', 'joao.silva@email.com', '11999999999'),
('Maria Oliveira', 'professor', 'maria.oliveira@universidade.com', '21888888888'),
('Carlos Pereira', 'pesquisador', 'carlos.pereira@instituto.br', '31777777777'),
('Ana Souza', 'aluno', 'ana.souza@mail.com', '41666666666'),
('Administrador da Biblioteca', 'administrador', 'admin@biblioteca.com', '51123456789'),
('Laura Martins', 'aluno', 'laura.martins@estudante.com', '19876543210');

-- Adicionando alguns empréstimos.
INSERT INTO Emprestimos (id_exemplar, id_usuario, data_devolucao_prevista) VALUES
((SELECT id FROM Exemplares WHERE codigo_barras = 'DOMC001'), 1, '2025-04-05'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'PEQP002'), 2, '2025-04-10'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'NINE003'), 1, '2025-04-15'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'SAPI004'), 3, '2025-04-20'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'ORGU005'), 4, '2025-04-25');
-- Adicionando alguns empréstimos já devolvidos para testar o histórico
INSERT INTO Emprestimos (id_exemplar, id_usuario, data_devolucao_prevista, data_devolucao_real) VALUES
((SELECT id FROM Exemplares WHERE codigo_barras = 'DOMC002'), 2, '2025-03-10', '2025-03-09'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'PEQP003'), 1, '2025-03-15', '2025-03-14'),
((SELECT id FROM Exemplares WHERE codigo_barras = 'NINE004'), 3, '2025-03-20', '2025-03-19');

--Inserindo algumas reservas.
INSERT INTO Reservas (id_exemplar, id_usuario) VALUES
((SELECT id FROM Exemplares WHERE codigo_barras = 'PEQP001'), 3),
((SELECT id FROM Exemplares WHERE codigo_barras = 'META001'), 1),
((SELECT id FROM Exemplares WHERE codigo_barras = 'SAPI001'), 2);

-- Inserindo alguns recursos digitais.
INSERT INTO RecursosDigitais (titulo, tipo, url, data_disponibilidade, descricao, numero_paginas, genero) VALUES
('Introdução à Programação em Python', 'e-book', 'https://exemplo.com/python', '2024-01-01', 'Um guia introdutório para a linguagem de programação Python, abordando desde os conceitos básicos até estruturas de dados e programação orientada a objetos.', 150, 'Programação'),
('Journal of Artificial Intelligence Research', 'periodico', 'https://jair.org/', NULL, 'Uma revista acadêmica revisada por pares dedicada à publicação de pesquisas de alta qualidade em inteligência artificial.', NULL, 'Inteligência Artificial'),
('Tese de Doutorado sobre Redes Neurais', 'tese', 'https://repositorio.com/tese_ia', '2023-05-15', 'Uma tese de doutorado explorando novas arquiteturas e algoritmos para redes neurais profundas.', 280, 'Ciência da Computação'),
('Base de Dados Scopus', 'base_de_dados', 'https://www.scopus.com/', NULL, 'Uma extensa base de dados bibliográfica contendo resumos e citações de artigos de periódicos revisados por pares.', NULL, 'Pesquisa Acadêmica'),
('Artigo sobre Machine Learning', 'periodico', 'https://example.com/ml_article', '2025-03-25', 'Um artigo científico que discute as últimas tendências e avanços em aprendizado de máquina.', 25, 'Aprendizado de Máquina');

--Inserindo dados em listas de leituras.
INSERT INTO ListasLeitura (nome, descricao, id_usuario) VALUES
('Leituras Obrigatórias - Literatura Brasileira', 'Livros essenciais da literatura brasileira para o curso de Letras.', NULL), -- Lista da biblioteca
('Tópicos Avançados em Inteligência Artificial', 'Materiais para estudo aprofundado em IA para alunos de pós-graduação.', NULL), -- Lista da biblioteca
('Meus Favoritos de Ficção Científica', 'Livros de ficção científica que mais gostei e recomendo.', 1), -- Lista do usuário João Silva
('Artigos Interessantes sobre História Moderna', 'Artigos que achei relevantes para minha pesquisa de doutorado.', 3), -- Lista do usuário Carlos Pereira
('Romances Clássicos Favoritos', 'Uma seleção dos meus romances clássicos preferidos.', 2); -- Lista da usuária Maria Oliveira

-- Inserindo títulos de livros nas listas de leitura.
INSERT INTO LivrosListasLeitura (id_lista, id_livro) VALUES
(1, 1), -- 'Dom Casmurro' em 'Leituras Obrigatórias - Literatura Brasileira'
(1, 2), -- 'O Pequeno Príncipe' em 'Leituras Obrigatórias - Literatura Brasileira'
(2, 4), -- 'Sapiens' em 'Tópicos Avançados em Inteligência Artificial'
(2, 5), -- 'A Metamorfose' em 'Tópicos Avançados em Inteligência Artificial'
(3, 3), -- '1984' em 'Meus Favoritos de Ficção Científica'
(3, 8), -- 'Neuromancer' em 'Meus Favoritos de Ficção Científica'
(4, 7), -- 'Cem Anos de Solidão' em 'Artigos Interessantes sobre História Moderna'
(5, 6), -- 'Orgulho e Preconceito' em 'Romances Clássicos Favoritos'
(5, 1); -- 'Dom Casmurro' em 'Romances Clássicos Favoritos'

```
-- Inserindo dados na tabela RecursosDigitaisListasLeitura.
INSERT INTO RecursosDigitaisListasLeitura (id_lista, id_recurso_digital) VALUES
(2, 1), -- 'Introdução à Programação em Python' em 'Tópicos Avançados em Inteligência Artificial'
(2, 3), -- 'Tese de Doutorado sobre Redes Neurais' em 'Tópicos Avançados em Inteligência Artificial'
(4, 5), -- 'Artigo sobre Machine Learning' em 'Artigos Interessantes sobre História Moderna'
(3, 1); -- 'Introdução à Programação em Python' em 'Meus Favoritos de Ficção Científica'

-- Inserindo dados na tabela HistoricoEmprestimos.
INSERT INTO HistoricoEmprestimos (id_exemplar, id_usuario, data_emprestimo, data_devolucao_prevista, data_devolucao_real)
SELECT id_exemplar, id_usuario, data_emprestimo, data_devolucao_prevista, data_devolucao_real
FROM Emprestimos
WHERE data_devolucao_real IS NOT NULL;

--1 Consulta: Calcular o número total de cópias disponíveis por gênero de livro.
SELECT l.genero, SUM(l.quantidade_disponivel) AS total_disponivel
FROM Livros l
GROUP BY l.genero
ORDER BY total_disponivel DESC;

--2 Consulta: Listar os usuários e o número de livros que eles têm atualmente emprestados.
SELECT u.nome, COUNT(e.id) AS numero_livros_emprestados
FROM Emprestimos e
LEFT JOIN Usuarios u ON e.id_usuario = u.id
WHERE e.data_devolucao_real IS NULL
GROUP BY u.nome
ORDER BY numero_livros_emprestados DESC;

--3 Consulta: encontrar o recurso digital com o maior número de páginas.
SELECT titulo, MAX(numero_paginas) AS max_paginas
FROM RecursosDigitais
WHERE numero_paginas IS NOT NULL
GROUP BY titulo
ORDER BY max_paginas DESC
LIMIT 1;

--4 Consulta: Encontrar o livro físico emprestado com o menor número de páginas .
SELECT l.titulo, MIN(l.numero_paginas) AS min_paginas
```

```
FROM Livros l
LEFT JOIN Exemplares ex ON l.id = ex.id_livro
LEFT JOIN Emprestimos e ON ex.id = e.id_exemplar
WHERE e.data_devolucao_real IS NULL AND l.numero_paginas IS NOT NULL
GROUP BY l.titulo
ORDER BY min_paginas ASC
LIMIT 1;

--5 Consulta: Encontrar os 5 livros mais emprestados.
SELECT l.titulo, COUNT(e.id) AS numero_emprestimos
FROM Livros l
INNER JOIN Exemplares ex ON l.id = ex.id_livro
INNER JOIN Emprestimos e ON ex.id = e.id_exemplar
GROUP BY l.titulo
ORDER BY numero_emprestimos DESC
LIMIT 5;

--6 Consulta: Listar todos os títulos de livros e o número de usuários que os favoritaram.
SELECT l.titulo, COUNT(f.id_usuario) AS numero_favoritos
FROM Favoritos f
RIGHT JOIN Livros l ON f.id_livro = l.id
GROUP BY l.titulo
ORDER BY numero_favoritos DESC;

--7 Consulta: Calcular a média de dias entre a data de empréstimo e a data de devolução prevista, considerando todos os empréstimos e juntando com informações dos usuários.
SELECT
    AVG(e.data_devolucao_prevista - e.data_emprestimo) AS media_dias_para_devolucao_prevista,
    u.nome AS nome_usuario
FROM Emprestimos e
FULL JOIN Usuarios u ON e.id_usuario = u.id
GROUP BY u.nome
ORDER BY u.nome;

--Consulta 8: Obras com maior tempo fora da biblioteca
SELECT e.id_exemplar, e.id_usuario, (e.data_devolucao_real - e.data_emprestimo) AS tempo
FROM Emprestimos e
WHERE e.data_devolucao_real IS NOT NULL
ORDER BY tempo DESC
LIMIT 5;
```

```
--9 Consulta: Estoque atual de exemplares disponíveis
SELECT COUNT(*) AS "Estoque atual de exemplares disponíveis"
FROM Exemplares WHERE status = 'disponivel';

--10 Consulta: Devoluções em atraso
SELECT COUNT(*) FROM Emprestimos
WHERE data_devolucao_real > data_devolucao_prevista;

- View 1: Livros físicos disponíveis (com título, autor e número de páginas).
CREATE VIEW vw_livros_disponiveis AS
SELECT l.titulo, l.autor, l.numero_paginas
FROM Livros l
WHERE l.quantidade_disponivel > 0;

-- View 2: Recursos digitais (com título, tipo e URL).
CREATE VIEW vw_recursos_digitais AS
SELECT titulo, tipo, url
FROM RecursosDigitais;

-- View 3: Usuários com empréstimos ativos (com nome e número de itens emprestados).
CREATE VIEW vw_usuarios_com_emprestimos AS
SELECT u.nome, COUNT(e.id) AS numero_emprestimos
FROM Usuarios u
INNER JOIN Emprestimos e ON u.id = e.id_usuario
WHERE e.data_devolucao_real IS NULL
GROUP BY u.nome;

--View 4: Os 5 livros mais emprestados
CREATE OR REPLACE VIEW vw_mais_emprestados AS
SELECT * FROM (
        SELECT
        'livro' AS tipo,
        l.id,
        l.titulo,
        l.descricao,
        l.genero,
        NULL AS url,
        COUNT(e.id) + COUNT(he.id) AS total_emprestimos
        FROM livros l
        LEFT JOIN emprestimos e ON e.id_exemplar IN (
        SELECT id FROM exemplares WHERE id_livro = l.id
        )
        LEFT JOIN historicoemprestimos he ON he.id_exemplar IN (
```

```
        SELECT id FROM exemplares WHERE id_livro = l.id
        )
        GROUP BY l.id
        ORDER BY total_emprestimos DESC
        LIMIT 5
) AS livros_mais_emprestados;

--View 5: Os 5 materiais mais favoritados
CREATE OR REPLACE VIEW vw_mais_favoritados AS
SELECT * FROM (
        SELECT
        'livro' AS tipo,
        l.id,
        l.titulo,
        l.descricao,
        l.genero,
        NULL AS url,
        COUNT(f.id) AS total_favoritos
        FROM livros l
        LEFT JOIN favoritos f ON f.id_livro = l.id
        GROUP BY l.id
        ORDER BY total_favoritos DESC
        LIMIT 5
) AS livros_favoritados

UNION ALL

SELECT * FROM (
        SELECT
        'recurso' AS tipo,
        r.id,
        r.titulo,
        r.descricao,
        r.genero,
        r.url,
        COUNT(f.id) AS total_favoritos
        FROM recursosdigitais r
        LEFT JOIN favoritos f ON f.id_recurso_digital = r.id
        GROUP BY r.id
        ORDER BY total_favoritos DESC
        LIMIT 5
) AS recursos_favoritados;

-- View 6: Últimos adicionados
```

```
CREATE OR REPLACE VIEW vw_ultimos_adicionados AS
SELECT * FROM (
	SELECT
	'livro' AS tipo,
	id,
	titulo,
	descricao,
	genero,
	NULL AS url
	FROM livros
	ORDER BY id DESC
	LIMIT 5
) AS livros_ultimos

UNION ALL

SELECT * FROM (
	SELECT
	'recurso' AS tipo,
	id,
	titulo,
	descricao,
	genero,
	url
	FROM recursosdigitais
	ORDER BY id DESC
	LIMIT 5
) AS recursos_ultimos;

-- Procedimento 1: Renovar o empréstimo de um livro.
CREATE OR REPLACE PROCEDURE sp_renovar_emprestimo(
	p_id_emprestimo INTEGER,
	p_nova_data_devolucao DATE
)
LANGUAGE plpgsql
AS $$
BEGIN
	-- Verificar se o empréstimo existe e ainda não foi devolvido
	IF NOT EXISTS (SELECT 1 FROM Emprestimos WHERE id =
p_id_emprestimo AND data_devolucao_real IS NULL) THEN
	RAISE NOTICE 'Empréstimo com ID % não encontrado ou já devolvido.',
p_id_emprestimo;
	RETURN;
	END IF;
```

```
        -- Atualizar a data de devolução prevista.
        UPDATE Emprestimos
        SET data_devolucao_prevista = p_nova_data_devolucao
        WHERE id = p_id_emprestimo;

        RAISE NOTICE 'Empréstimo com ID % renovado para %.', p_id_emprestimo,
p_nova_data_devolucao;
END;
$$;

-- Procedimento 2: Registrar a reserva de um livro.
CREATE OR REPLACE PROCEDURE sp_reservar_livro(
        p_id_usuario INTEGER,
        p_id_exemplar INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
        -- Verificar se o usuário existe
        IF NOT EXISTS (SELECT 1 FROM Usuarios WHERE id = p_id_usuario)
THEN
        RAISE NOTICE 'Usuário com ID % não encontrado.', p_id_usuario;
        RETURN;
        END IF;

        -- Verificar se o exemplar existe.
        IF NOT EXISTS (SELECT 1 FROM Exemplares WHERE id = p_id_exemplar)
THEN
        RAISE NOTICE 'Exemplar com ID % não encontrado.', p_id_exemplar;
        RETURN;
        END IF;

        -- Verificar se o exemplar já está reservado por este usuário.
        IF EXISTS (SELECT 1 FROM Reservas WHERE id_usuario = p_id_usuario
AND id_exemplar = p_id_exemplar) THEN
        RAISE NOTICE 'Exemplar com ID % já está reservado pelo usuário com ID
%.', p_id_exemplar, p_id_usuario;
        RETURN;
        END IF;

        -- Verificar se o exemplar está disponível (se estiver, talvez o usuário devesse
emprestar diretamente).
```

```
    IF EXISTS (SELECT 1 FROM Exemplares WHERE id = p_id_exemplar AND
status = 'disponivel') THEN
    RAISE NOTICE 'O exemplar com ID % está disponível para empréstimo
imediato.', p_id_exemplar;
    -- Você pode optar por não permitir a reserva se o livro estiver disponível.
    -- RETURN;
    END IF;

    -- Registrar a reserva.
    INSERT INTO Reservas (id_usuario, id_exemplar, data_reserva)
    VALUES (p_id_usuario, p_id_exemplar, CURRENT_DATE);

    RAISE NOTICE 'Reserva para o exemplar com ID % realizada pelo usuário
com ID % em %.', p_id_exemplar, p_id_usuario, CURRENT_DATE;
END;
$$;

-- Gatilho 1: Ao Criar um Empréstimo o status sempre é 'pendente'.
CREATE OR REPLACE FUNCTION fn_emprestimo_pendente()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE emprestimos
    SET status = 'pendente'
    WHERE id = NEW.id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_definir_emprestimo_pendente
AFTER INSERT ON emprestimos
FOR EACH ROW
EXECUTE FUNCTION fn_emprestimo_pendente();

-- Gatilho 2: Quando o Administrador Confirma a Retirada,
-- o status do empréstimo muda de 'pendente' para 'ativo'.
-- O exemplar muda para 'emprestado'.
-- O estoque do livro (quantidade_disponivel) é decrementado em 1.
CREATE OR REPLACE FUNCTION fn_ativar_emprestimo()
RETURNS TRIGGER AS $$
DECLARE
    livro_id INT;
BEGIN
    -- Atualizar status do exemplar
```

```
        UPDATE exemplares
        SET status = 'emprestado'
        WHERE id = NEW.id_exemplar;

        -- Atualizar estoque do livro
        SELECT id_livro INTO livro_id FROM exemplares WHERE id =
NEW.id_exemplar;
        UPDATE livros
        SET quantidade_disponivel = quantidade_disponivel - 1
        WHERE id = livro_id;

        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_emprestimo_ativo
AFTER UPDATE OF status ON emprestimos
FOR EACH ROW
WHEN (NEW.status = 'ativo' AND OLD.status = 'pendente')
EXECUTE FUNCTION fn_ativar_emprestimo();

-- Gatilho 3: Na Devolução status do empréstimo muda para 'concluido'.
-- O exemplar volta para 'disponivel'.
-- O estoque do livro é incrementado em 1.
CREATE OR REPLACE FUNCTION fn_processar_devolucao()
RETURNS TRIGGER AS $$
DECLARE
        livro_id INT;
BEGIN
        SELECT id_livro INTO livro_id FROM exemplares WHERE id =
NEW.id_exemplar;

        -- Atualizar estoque
        UPDATE livros
        SET quantidade_disponivel = quantidade_disponivel + 1
        WHERE id = livro_id;

        -- Atualizar status do exemplar
        UPDATE exemplares
        SET status = 'disponivel'
        WHERE id = NEW.id_exemplar;

        -- Gerar histórico
        INSERT INTO HistoricoEmprestimos (
```

```
        id_exemplar,
        id_usuario,
        data_emprestimo,
        data_devolucao_prevista,
        data_devolucao_real,
        data_registro_historico
        )
        VALUES (
        NEW.id_exemplar,
        NEW.id_usuario,
        NEW.data_emprestimo,
        NEW.data_devolucao_prevista,
        NEW.data_devolucao_real,
        CURRENT_TIMESTAMP
        );

        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_emprestimo_concluido
AFTER UPDATE OF data_devolucao_real ON emprestimos
FOR EACH ROW
WHEN (NEW.data_devolucao_real IS NOT NULL AND OLD.data_devolucao_real IS
NULL)
EXECUTE FUNCTION fn_processar_devolucao();
```