```sql
--1. Script de Criação de Tabelas com Restrições
-- Criação da tabela Livro
CREATE TABLE Livro (
    livro_id SERIAL PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    autor VARCHAR(255),
    isbn VARCHAR(13) UNIQUE,  -- ISBN deve ser único
    categoria VARCHAR(100) NOT NULL CHECK (categoria IN ('Ficção', 'Romance', 'Não
Ficção'))  -- Categoria restrita a valores específicos
);

-- Criação da tabela Usuario
CREATE TABLE Usuario (
    usuario_id SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    endereco VARCHAR(255),
    data_nascimento DATE,
    telefone VARCHAR(20) UNIQUE  -- Telefone deve ser único
);

-- Criação da tabela Funcionario
CREATE TABLE Funcionario (
    funcionario_id SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    cargo VARCHAR(100) NOT NULL
);

-- Criação da tabela Emprestimo
CREATE TABLE Emprestimo (
    emprestimo_id SERIAL PRIMARY KEY,
    usuario_id INT REFERENCES Usuario(usuario_id) ON DELETE CASCADE,  -- Chave
estrangeira para Usuario com exclusão em cascata
    livro_id INT REFERENCES Livro(livro_id) ON DELETE CASCADE,
    funcionario_id INT REFERENCES Funcionario(funcionario_id),
    data_emprestimo DATE NOT NULL,
    data_devolucao DATE,
    CONSTRAINT data_devolucao_check CHECK (data_devolucao IS NULL OR data_devolucao
 >= data_emprestimo)  -- Data de devolução deve ser maior ou igual à data de
empréstimo, se especificada
);


-- Criação da tabela Reserva
CREATE TABLE Reserva (
    reserva_id SERIAL PRIMARY KEY,
    usuario_id INT REFERENCES Usuario(usuario_id) ON DELETE CASCADE,
    livro_id INT REFERENCES Livro(livro_id) ON DELETE CASCADE,
    funcionario_id INT REFERENCES Funcionario(funcionario_id),
    data_reserva DATE NOT NULL
);


-- 2. Criação de índices
CREATE INDEX idx_usuario_nome ON Usuario(nome);  -- Índice no nome dos usuários
CREATE INDEX idx_livro_categoria ON Livro(categoria);  -- Índice na categoria dos
livros
```

```sql
CREATE INDEX idx_livro_titulo ON Livro(titulo); -- Índice no título dos livros

-- 3. População das tabelas

-- Inserção de dados na tabela Livro
INSERT INTO Livro (titulo, autor, isbn, categoria) VALUES
('Dom Casmurro', 'Machado de Assis', '9788535914848', 'Ficção'),
('Memórias Póstumas de Brás Cubas', 'Machado de Assis', '9788535914855', 'Ficção'),
('O Guarani', 'José de Alencar', '9788520934012', 'Romance'),
('Iracema', 'José de Alencar', '9788520934029', 'Romance'),
('Capitães da Areia', 'Jorge Amado', '9788520935019', 'Ficção'),
('Grande Sertão: Veredas', 'João Guimarães Rosa', '9788520935026', 'Ficção'),
('A Moreninha', 'Joaquim Manuel de Macedo', '9788520935033', 'Romance'),
('Senhora', 'José de Alencar', '9788520935040', 'Romance');

-- Inserção de dados na tabela Usuario
INSERT INTO Usuario (nome, endereco, data_nascimento) VALUES
('Alice Silva', 'Rua das Flores, 123', '2000-01-15'),
('Marco Santos', 'Avenida das Acácias, 456', '1998-05-22'),
('Carlos Oliveira', 'Rua das Palmeiras, 789', '2004-03-30'),
('Daniela Almeida', 'Avenida Central, 101', '1999-12-11');

-- Inserção de dados na tabela Funcionario
INSERT INTO Funcionario (nome, cargo) VALUES
('João Pereira', 'Bibliotecário'),
('Maria Oliveira', 'Assistente de Biblioteca'),
('Pedro Lima', 'Gerente de Biblioteca'),
('Ana Souza', 'Atendente');


--4. Criação de Funções

-- Função para obter livros por autor
CREATE FUNCTION obter_livros_por_autor(nome_autor VARCHAR) RETURNS TABLE(livroid INT
, titulo_livro VARCHAR) AS $$
BEGIN
    RETURN QUERY SELECT livro_id, titulo FROM Livro WHERE autor = nome_autor;
END;
$$ LANGUAGE plpgsql;

drop function obter_livros_por_autor(nome_autor VARCHAR);

-- Função para contar o total de empréstimos por usuário
CREATE FUNCTION total_emprestimos_por_usuario(usuarioid INT) RETURNS INT AS $$
BEGIN
    RETURN (SELECT COUNT(*) FROM Emprestimo WHERE usuario_id = usuario_id);
END;
$$ LANGUAGE plpgsql;

drop function total_emprestimos_por_usuario(usuarioid INT);

-- Função para listar livros emprestados por usuário
CREATE FUNCTION livros_emprestados_por_usuario(usuarioid INT) RETURNS TABLE(titulo
VARCHAR, data_emprestimo DATE, data_devolucao DATE) AS $$
BEGIN
    RETURN QUERY
```

```
    SELECT l.titulo, e.data_emprestimo, e.data_devolucao
    FROM Emprestimo e
    JOIN Livro l ON e.livro_id = l.livro_id
    WHERE e.usuario_id = usuarioid;
END;
$$ LANGUAGE plpgsql;

drop function livros_emprestados_por_usuario(usuarioid INT);

--5. Criação de VIEW

-- Visão para listar livros emprestados
CREATE VIEW livros_emprestados AS
SELECT u.nome AS usuario, l.titulo, e.data_emprestimo, e.data_devolucao
FROM Emprestimo e
JOIN Usuario u ON e.usuario_id = u.usuario_id
JOIN Livro l ON e.livro_id = l.livro_id;

-- Visão para listar reservas ativas
CREATE VIEW reservas_ativas AS
SELECT u.nome AS usuario, l.titulo, r.data_reserva
FROM Reserva r
JOIN Usuario u ON r.usuario_id = u.usuario_id
JOIN Livro l ON r.livro_id = l.livro_id
WHERE r.data_reserva >= CURRENT_DATE;

-- Visão para listar livros disponíveis
CREATE VIEW livros_disponiveis AS
SELECT l.titulo, l.autor, l.categoria
FROM Livro l
LEFT JOIN Emprestimo e ON l.livro_id = e.livro_id AND e.data_devolucao IS NULL
WHERE e.emprestimo_id IS NULL;

-- Criação da tabela de auditoria para a tabela Emprestimo
CREATE TABLE Auditoria_Emprestimo (
    auditoria_id SERIAL PRIMARY KEY,
    emprestimo_id INT REFERENCES Emprestimo(emprestimo_id) ON DELETE CASCADE,
    operacao CHAR(1) CHECK (operacao IN ('I', 'U', 'D')),  -- Operação deve ser I (
Inserir), U (Atualizar) ou D (Deletar)
    data_operacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    funcionario_id INT REFERENCES Funcionario(funcionario_id)
);

-- 6. Criação de Triggers para Tratamento de Eventos e Auditoria

-- Função para log de inserção na tabela Emprestimo
CREATE FUNCTION log_inserir_emprestimo() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao, data_operacao,
funcionario_id)
    VALUES (NEW.emprestimo_id, 'I', CURRENT_TIMESTAMP, NEW.funcionario_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger para logar inserções na tabela Emprestimo
```

```sql
CREATE TRIGGER trigger_inserir_emprestimo
AFTER INSERT ON Emprestimo
FOR EACH ROW
EXECUTE FUNCTION log_inserir_emprestimo();

-- Função para log de atualização na tabela Emprestimo
CREATE FUNCTION log_atualizar_emprestimo() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao, data_operacao,
funcionario_id)
    VALUES (NEW.emprestimo_id, 'U', CURRENT_TIMESTAMP, NEW.funcionario_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger para logar atualizações na tabela Emprestimo
CREATE TRIGGER trigger_atualizar_emprestimo
AFTER UPDATE ON Emprestimo
FOR EACH ROW
EXECUTE FUNCTION log_atualizar_emprestimo();

-- Função para log de deleção na tabela Emprestimo
CREATE FUNCTION log_deletar_emprestimo() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO Auditoria_Emprestimo (emprestimo_id, operacao, data_operacao,
funcionario_id)
    VALUES (OLD.emprestimo_id, 'D', CURRENT_TIMESTAMP, OLD.funcionario_id);
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

-- Trigger para logar deleções na tabela Emprestimo
CREATE TRIGGER trigger_deletar_emprestimo
AFTER DELETE ON Emprestimo
FOR EACH ROW
EXECUTE FUNCTION log_deletar_emprestimo();
```