# Relatório da Aula Prática: Programação e Desenvolvimento de Banco de Dados

## 1. Introdução

Este relatório documenta a execução da aula prática de Programação e Desenvolvimento de Banco de Dados, cujo objetivo principal foi criar um banco de dados utilizando a linguagem SQL e realizar operações de manipulação e acesso aos dados. A atividade focou na implementação de um sistema de gestão de contas a receber, abrangendo a criação da estrutura do banco de dados, inserção de dados de exemplo e a elaboração de consultas específicas.

## 2. Métodos

A aula prática foi dividida em três etapas principais, conforme o roteiro fornecido. Todas as etapas foram realizadas em um ambiente Linux utilizando o MySQL Community Server e o MySQL Workbench (simulado via linha de comando para a execução dos scripts SQL).

### 2.1. Criação da Estrutura do Banco de Dados

A primeira etapa consistiu na criação do banco de dados Loja e de suas tabelas (Estado, Municipio, Cliente, ContaReceber) utilizando comandos DDL (Data Definition Language). O modelo físico foi implementado seguindo o diagrama entidade-relacionamento (DER) pré-definido no roteiro. Foi dada atenção especial à ordem de criação das tabelas para respeitar as dependências de chaves estrangeiras. O script create\_database.sql foi utilizado para esta finalidade.

CREATE DATABASE IF NOT EXISTS Loja;

USE Loja;

CREATE TABLE IF NOT EXISTS Estado (

ID INT PRIMARY KEY,

Nome VARCHAR(80) NOT NULL,

UF CHAR(2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Municipio (

ID INT PRIMARY KEY,

Nome VARCHAR(80) NOT NULL,

Estado\_ID INT NOT NULL,

FOREIGN KEY (Estado\_ID) REFERENCES Estado(ID)

);

CREATE TABLE IF NOT EXISTS Cliente (

ID INT PRIMARY KEY,

Nome VARCHAR(80) NOT NULL,

CPF CHAR(11) NOT NULL UNIQUE,

Celular VARCHAR(10),

Endereco VARCHAR(100),

EndNumero VARCHAR(10),

EndCEP CHAR(8),

Municipio\_ID INT NOT NULL,

FOREIGN KEY (Municipio\_ID) REFERENCES Municipio(ID)

);

CREATE TABLE IF NOT EXISTS ContaReceber (

ID INT PRIMARY KEY,

Cliente\_ID INT NOT NULL,

DataVencimento DATE NOT NULL,

Valor DECIMAL(18,2) NOT NULL,

Situacao ENUM(\'1\', \'2\', \'3\') NOT NULL COMMENT \'1=Paga, 2=Cancelada, 3=Pendente\',

FOREIGN KEY (Cliente\_ID) REFERENCES Cliente(ID)

);

### 2.2. Inserção de Dados

Após a criação da estrutura, a segunda etapa envolveu a inserção de dados de exemplo nas tabelas. Para isso, foi elaborado o script inserir.sql, contendo comandos DML (Data Manipulation Language) para popular as tabelas Estado, Municipio, Cliente e ContaReceber com informações fictícias. O script garantiu a inserção de pelo menos três registros por tabela.

USE Loja;

-- Inserir dados na tabela Estado

INSERT INTO Estado (ID, Nome, UF) VALUES

(1, \'São Paulo\', \'SP\'),

(2, \'Rio de Janeiro\', \'RJ\'),

(3, \'Minas Gerais\', \'MG\');

-- Inserir dados na tabela Municipio

INSERT INTO Municipio (ID, Nome, Estado\_ID) VALUES

(101, \'São Paulo\', 1),

(102, \'Campinas\', 1),

(103, \'Rio de Janeiro\', 2),

(104, \'Niterói\', 2),

(105, \'Belo Horizonte\', 3),

(106, \'Uberlândia\', 3);

-- Inserir dados na tabela Cliente

INSERT INTO Cliente (ID, Nome, CPF, Celular, Endereco, EndNumero, EndCEP, Municipio\_ID) VALUES

(1, \'João Silva\', \'11122233344\', \'1198765432\', \'Rua A\', \'100\', \'01000000\', 101),

(2, \'Maria Oliveira\', \'22233344455\', \'2199887766\', \'Av. B\', \'250\', \'20000000\', 103),

(3, \'Pedro Souza\', \'33344455566\', \'3197766554\', \'Rua C\', \'50\', \'30000000\', 105);

-- Inserir dados na tabela ContaReceber

INSERT INTO ContaReceber (ID, Cliente\_ID, DataVencimento, Valor, Situacao) VALUES

(1, 1, \'2025-10-30\', 150.75, \'1\'),

(2, 1, \'2025-11-15\', 200.00, \'3\'),

(3, 2, \'2025-10-25\', 50.50, \'1\'),

(4, 3, \'2025-12-01\', 300.00, \'3\');

### 2.3. Elaboração de Consultas (VIEW)

A terceira etapa envolveu a criação de uma visão (VIEW) para consultar as contas a receber que ainda não foram pagas. O script consulta.sql foi desenvolvido utilizando comandos DQL (Data Query Language) para criar a VIEW ContasNaoPagas, que retorna o ID da conta, nome e CPF do cliente, data de vencimento e valor da conta para todas as contas com Situacao = '3' (pendente).

USE Loja;

CREATE OR REPLACE VIEW ContasNaoPagas AS

SELECT

cr.ID AS ID\_Conta,

c.Nome AS Nome\_Cliente,

c.CPF AS CPF\_Cliente,

cr.DataVencimento AS Data\_Vencimento,

cr.Valor AS Valor\_Conta

FROM

ContaReceber cr

JOIN

Cliente c ON cr.Cliente\_ID = c.ID

WHERE

cr.Situacao = \'3\';

## 3. Resultados

Após a execução dos scripts SQL, o banco de dados Loja foi criado com sucesso, e as tabelas foram populadas com os dados especificados. A VIEW ContasNaoPagas também foi criada e testada, retornando as informações esperadas.

### 3.1. Tabelas Criadas

As seguintes tabelas foram criadas no banco de dados Loja:

+----------------+

| Tables\_in\_Loja |

+----------------+

| Cliente |

| ContaReceber |

| Estado |

| Municipio |

+----------------+

### 3.2. Consulta de Contas Não Pagas

A consulta à VIEW ContasNaoPagas retornou as seguintes contas pendentes:

+----------+--------------+-------------+-----------------+-------------+

| ID\_Conta | Nome\_Cliente | CPF\_Cliente | Data\_Vencimento | Valor\_Conta |

+----------+--------------+-------------+-----------------+-------------+

| 2 | João Silva | 11122233344 | 2025-11-15 | 200.00 |

| 4 | Pedro Souza | 33344455566 | 2025-12-01 | 300.00 |

+----------+--------------+-------------+-----------------+-------------+

## 4. Conclusão

A aula prática foi concluída com êxito, demonstrando a capacidade de criar e gerenciar um banco de dados relacional utilizando MySQL e comandos SQL. Todas as etapas do roteiro foram cumpridas, desde a definição da estrutura do banco de dados e suas tabelas, passando pela inserção de dados, até a criação de uma visão para consultas específicas. A atividade proporcionou uma compreensão prática dos conceitos de DDL e DML, bem como a aplicação de chaves primárias e estrangeiras, tipos de dados e restrições de integridade. A criação da VIEW também ilustrou a capacidade de simplificar consultas complexas e melhorar a segurança do acesso aos dados.