```
-- Criação das tabelas base do sistema escolar
CREATE TABLE Curso (
    CursoID INT PRIMARY KEY, -- Identificador único do curso
    Nome VARCHAR(100),       -- Nome do curso
    ValorMensalidade DECIMAL(10,2) -- Valor da mensalidade
);

CREATE TABLE Aluno (
    AlunoID INT PRIMARY KEY IDENTITY, -- Identificador único do aluno (auto
incremento)
    Nome VARCHAR(100),                -- Nome do aluno
    CPF VARCHAR(11),                  -- CPF do aluno
    CursoID INT FOREIGN KEY REFERENCES Curso(CursoID) -- Curso relacionado ao
aluno
);

-- Inserção de cursos no sistema
INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (1, 'Análise e
Desenvolvimento de Sistemas', 850.00);
INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (2, 'Administração',
780.00);
INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (3, 'Engenharia
Civil', 1200.00);
INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (4, 'Direito',
1100.00);
INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (5, 'Enfermagem',
900.00);

-- Inserção de alunos já matriculados
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Mariana Lopes Silva',
'12345678901', 1);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('João Pedro Almeida',
'23456789012', 2);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Beatriz Costa Rocha',
'34567890123', 1);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Lucas Henrique Sousa',
'45678901234', 3);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Ana Clara Fernandes',
'56789012345', 4);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Carlos Eduardo Braga',
'67890123456', 5);
INSERT INTO Aluno (Nome, CPF, CursoID) VALUES ('Juliana Martins', '78901234567',
3);
```

```
-- TRANSACTION

BEGIN TRANSACTION

INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (9, 'TESTE 9', 900.00)

ROLLBACK
COMMIT

select * from Curso

-- TRY CATCH
BEGIN TRY

     INSERT INTO Curso (CursoID, Nome, ValorMensalidade) VALUES (9, 'TESTE 9',
900.00)

END TRY
BEGIN CATCH
        ROLLBACK
        PRINT 'Erro ao inserir curso: ' + ERROR_MESSAGE()
END CATCH


BEGIN TRY

     RAISERROR('mensagem de erro customizada', 16, 1)

END TRY
BEGIN CATCH
        ROLLBACK
        PRINT 'Erro ao inserir curso: ' + ERROR_MESSAGE()
END CATCH

--RAISERROR('mensagem', severidade, estado)
--mensagem: Texto a ser exibido
--severidade: Nível do erro (11 a 16 para erros definidos pelo usuário)
--estado: Número de controle (use 1)



-- Procedure 1: Inserir um novo aluno
CREATE PROCEDURE InserirAluno
    @Nome VARCHAR(100),
    @CPF VARCHAR(11),
    @CursoID INT
```

```
AS
BEGIN
    INSERT INTO Aluno (Nome, CPF, CursoID)
    VALUES (@Nome, @CPF, @CursoID)
END

-- Procedure 2: Atualizar o valor da mensalidade de um curso
CREATE PROCEDURE AtualizarMensalidadeCurso
    @CursoID INT,
    @NovoValor DECIMAL(10,2)
AS
BEGIN
    UPDATE Curso
    SET ValorMensalidade = @NovoValor
    WHERE CursoID = @CursoID
END

-- Procedure 3: Remover um aluno com base no ID
CREATE PROCEDURE RemoverAluno
    @AlunoID INT
AS
BEGIN
    DELETE FROM Aluno WHERE AlunoID = @AlunoID
END

-- Procedure 4: Listar alunos matriculados em um determinado curso
CREATE PROCEDURE ListarAlunosPorCurso
    @CursoID INT
AS
BEGIN
    SELECT
        A.AlunoID,
        A.Nome AS NomeAluno,
        A.CPF,
        C.Nome AS NomeCurso,
        C.ValorMensalidade
    FROM Aluno A
    INNER JOIN Curso C ON A.CursoID = C.CursoID
    WHERE A.CursoID = @CursoID
END

-- Procedure 5: Matricular aluno com transação e validação
CREATE PROCEDURE MatricularAluno
    @Nome VARCHAR(100),
    @CPF VARCHAR(11),
    @CursoID INT
AS
```

```
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION

        -- Verifica se o curso existe
        IF NOT EXISTS (SELECT 1 FROM Curso WHERE CursoID = @CursoID)
        BEGIN
            RAISERROR('Curso informado não existe.', 16, 1)
        END

        -- Realiza a inserção
        INSERT INTO Aluno (Nome, CPF, CursoID)
        VALUES (@Nome, @CPF, @CursoID)

        COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        PRINT 'Erro ao matricular aluno: ' + ERROR_MESSAGE()
    END CATCH
END

-- Procedure 6: Inserir curso com validação da mensalidade
CREATE PROCEDURE InserirCursoComValidacao
    @Nome VARCHAR(100),
    @ValorMensalidade DECIMAL(10,2)
AS
BEGIN
    BEGIN TRY
        -- Verifica se a mensalidade é válida
        IF @ValorMensalidade <= 0
        BEGIN
            RAISERROR('Valor da mensalidade deve ser maior que zero.', 16, 1)
        END

        -- Insere o curso
        INSERT INTO Curso (Nome, ValorMensalidade)
        VALUES (@Nome, @ValorMensalidade)
    END TRY
    BEGIN CATCH
        PRINT 'Erro: ' + ERROR_MESSAGE()
    END CATCH
END

-- Procedure 7: Matricular aluno com verificação de vagas no curso
-- Procedure 7: Matricular aluno com verificação de vagas no curso
CREATE PROCEDURE MatricularAlunoComLimite
```

```
    @Nome VARCHAR(100),
    @CPF VARCHAR(11),
    @CursoID INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION

        -- Verifica se o curso existe
        IF NOT EXISTS (SELECT 1 FROM Curso WHERE CursoID = @CursoID)
        BEGIN
            RAISERROR('Curso informado não existe.', 16, 1)
        END

        -- Verifica se ainda há vagas disponíveis (máximo 3 alunos)
        DECLARE @TotalAlunos INT
        SELECT @TotalAlunos = COUNT(*) FROM Aluno WHERE CursoID = @CursoID

        IF @TotalAlunos >= 3
        BEGIN
            RAISERROR('Curso atingiu o limite de vagas.', 16, 1)
        END

        -- Realiza a matrícula
        INSERT INTO Aluno (Nome, CPF, CursoID)
        VALUES (@Nome, @CPF, @CursoID)

        COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        PRINT 'Erro ao matricular aluno: ' + ERROR_MESSAGE()
    END CATCH
END

EXEC MatricularAlunoComLimite 'Renata Oliveira', '98765432100', 1
```