

## BANCO DE DADOS

### Trabalho – Relatório

<b>Curso:</b>	Bacharelado em Engenharia de Software
<b>Aluno(a):</b>	Carlos Henrique Monnerat Quintanilha
<b>RU:</b>	4328237

#### 1. 1ª Etapa – Modelagem

**Pontuação:** 25 pontos.

Dadas as regras de negócio abaixo listadas, referentes ao estudo de caso de uma companhia aérea, elabore o Modelo Entidade-Relacionamento (MER), isto é, o modelo conceitual.

O Modelo Entidade-Relacionamento (MER) deve contemplar os seguintes itens:

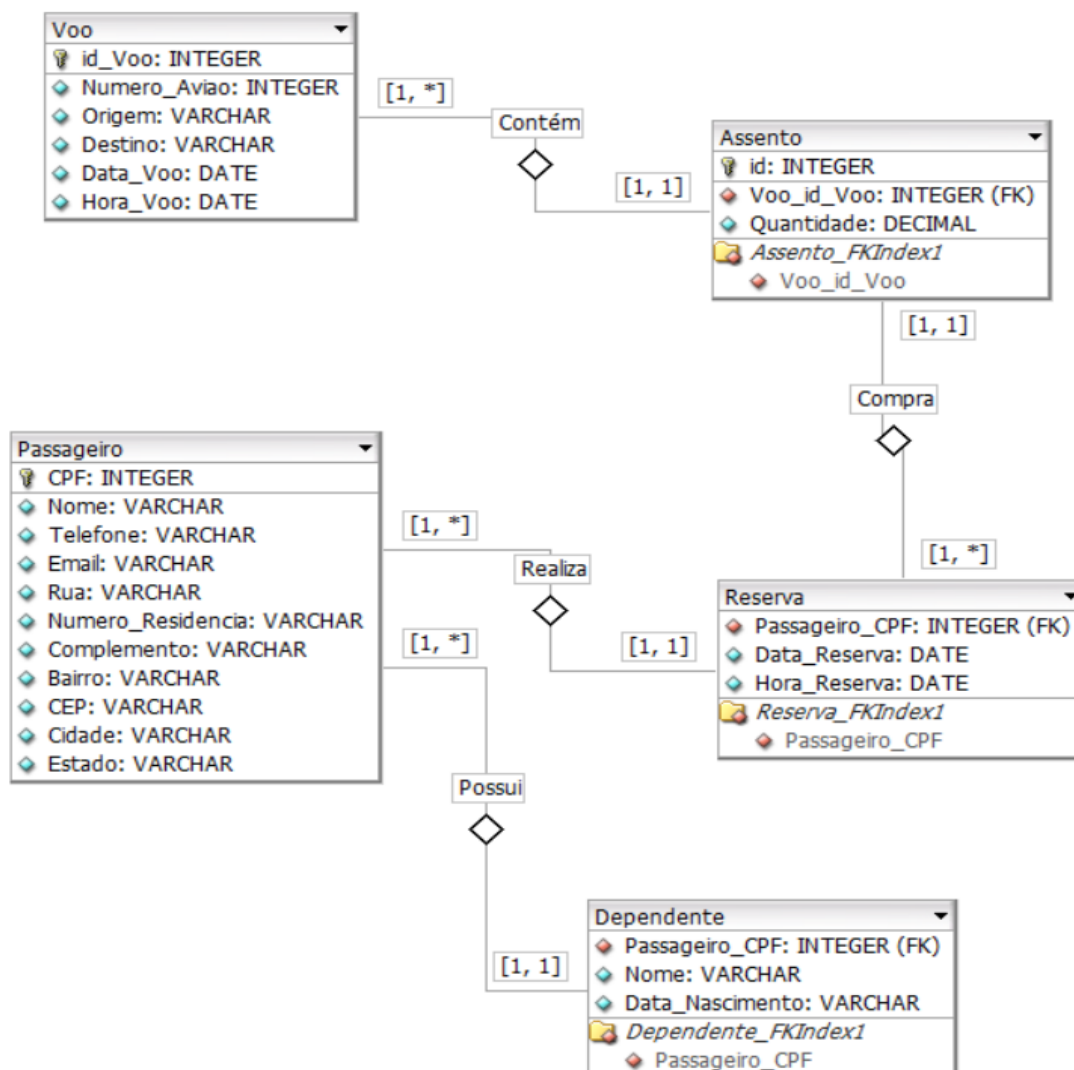
- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

Uma companhia aérea necessita controlar os dados de seus voos. Para isso, contratou um profissional de Banco de Dados, a fim de modelar o Banco de Dados que armazenará os dados dos voos.

As regras de negócio são:

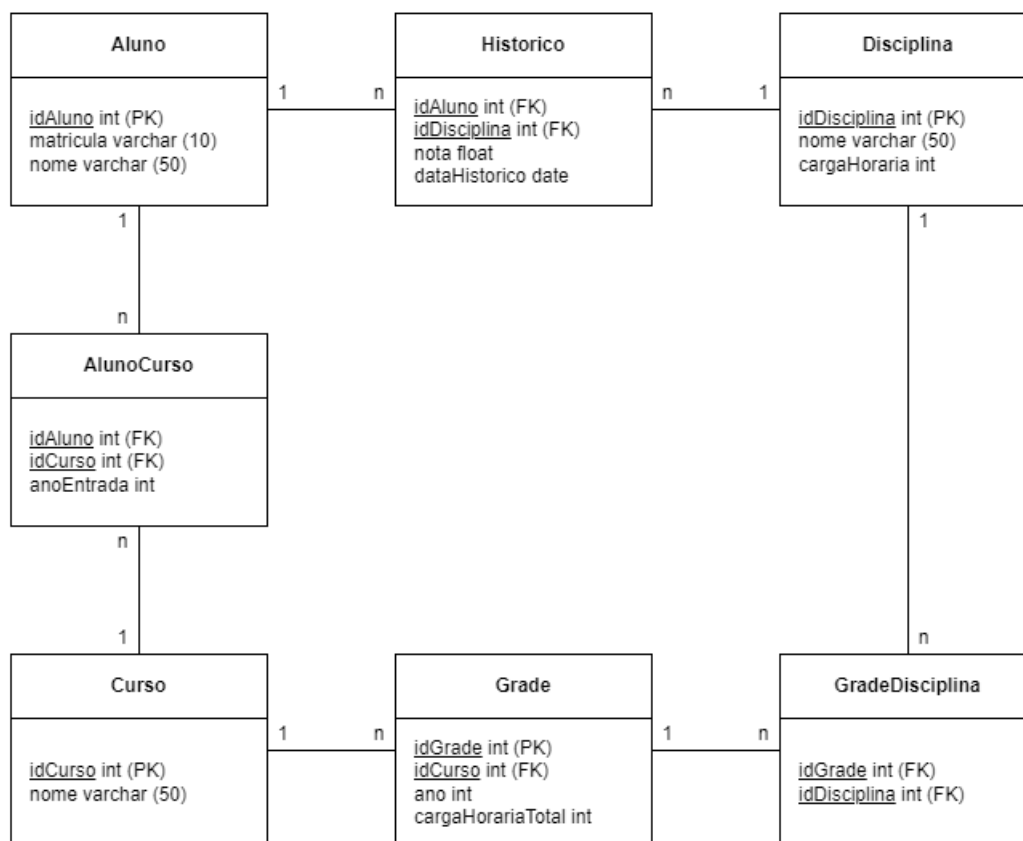
- Voo – Deverão ser armazenados os seguintes dados: identificação do voo, número do avião, cidade de origem, cidade de destino, data do voo e hora do voo;
- Assento – Deverão ser armazenados os seguintes dados: identificação do assento e quantidade;

- Passageiro – Deverão ser armazenados os seguintes dados: CPF, nome, telefone, e-mail e endereço (rua, número, complemento, bairro, CEP, cidade e estado);
- Dependente – Deverão ser armazenados os seguintes dados: nome e data de nascimento;
- Um voo pode ter zero ou vários assentos, assim como zero ou vários assentos pertencem a um voo;
- Um passageiro pode ter zero ou várias reservas de assentos, assim como zero ou várias reservas de assentos pertencem a um passageiro;
- Um passageiro pode ter zero ou vários dependentes, assim como zero ou vários dependentes são de um passageiro;
- Da reserva deverão ser armazenados os seguintes dados: data da reserva e hora da reserva.



## 2. 2ª Etapa – Implementação

Considere o seguinte Modelo Relacional (modelo lógico), referente ao estudo de caso de uma faculdade:



Com base no Modelo Relacional dado e utilizando a *Structured Query Language* (SQL), no MySQL Workbench, implemente o que se pede.

**Observação:** Para testar o Banco de Dados após a implementação, utilize os comandos contidos no arquivo “Trabalho – Populando o Banco de Dados” para popular as tabelas. Tal arquivo contém todos os comandos de inserção dos dados (fictícios) necessários para a realização dos testes.

**Pontuação:** 25 pontos.

1. Implemente um Banco de Dados chamado “Faculdade”. Após, implemente as tabelas, conforme o Modelo Relacional dado, observando as chaves primárias e as chaves estrangeiras. Todos os campos, de todas as tabelas, não podem ser nulos (*not null*).

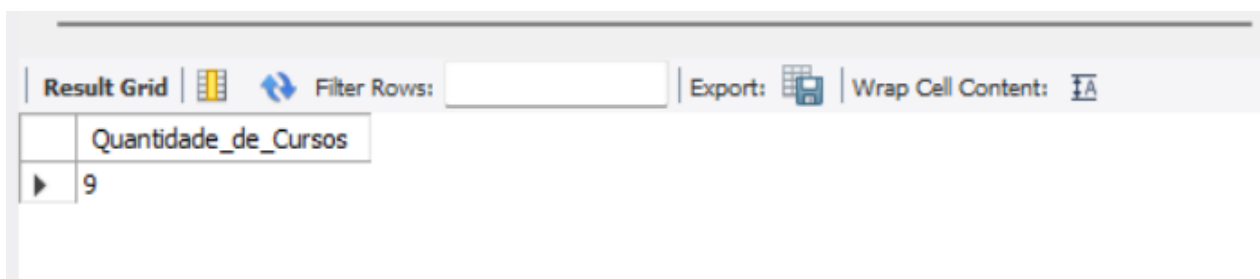
```
CREATE DATABASE Faculdade;
USE Faculdade;
CREATE TABLE Aluno (
    idAluno INT NOT NULL PRIMARY KEY,
    matricula VARCHAR(10) NOT NULL,
    nome VARCHAR(50) NOT NULL
);
CREATE TABLE Curso (
    idCurso INT NOT NULL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL
);
CREATE TABLE Disciplina (
    idDisciplina INT NOT NULL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    cargaHoraria INT NOT NULL
);
CREATE TABLE AlunoCurso (
    idAluno INT NOT NULL,
    idCurso INT NOT NULL,
    anoEntrada INT NOT NULL,
    PRIMARY KEY (idAluno, idCurso),
    FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),
    FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)
);
CREATE TABLE Historico (
    idAluno INT NOT NULL,
    idDisciplina INT NOT NULL,
```

```
nota FLOAT NOT NULL,  
dataHistorico DATE NOT NULL,  
PRIMARY KEY (idAluno, idDisciplina),  
FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),  
FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)  
);  
CREATE TABLE Grade (  
idGrade INT NOT NULL PRIMARY KEY,  
idCurso INT NOT NULL,  
ano INT NOT NULL,  
cargaHorariaTotal INT NOT NULL,  
FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)  
);  
CREATE TABLE GradeDisciplina (  
idGrade INT NOT NULL,  
idDisciplina INT NOT NULL,  
PRIMARY KEY (idGrade, idDisciplina),  
FOREIGN KEY (idGrade) REFERENCES Grade(idGrade),  
FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)  
);
```

**Pontuação:** 10 pontos.

2. Implemente uma consulta para listar o quantitativo de cursos existentes.

```
SELECT COUNT(*) AS Quantidade_de_Cursos FROM Curso;
```

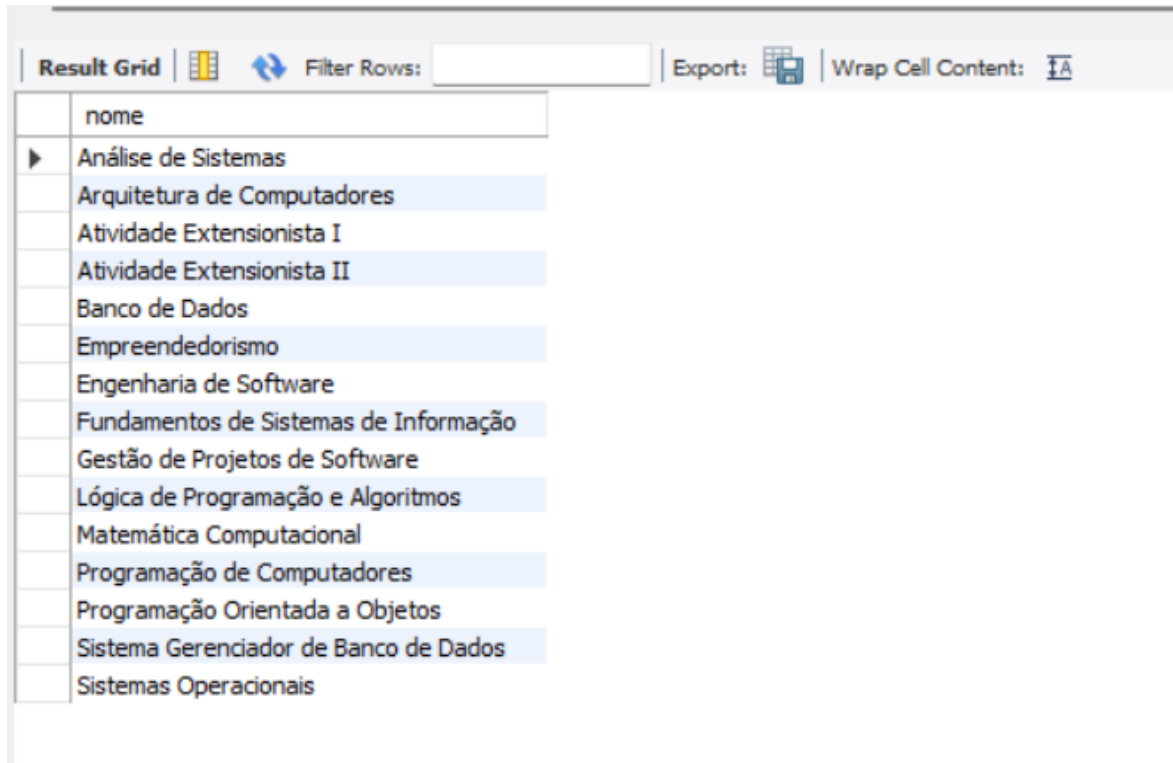


	Quantidade_de_Cursos
▶	9

**Pontuação:** 10 pontos.

3. Implemente uma consulta para listar o nome das disciplinas existentes.

**SELECT nome FROM Disciplina;**



	nome
▶	Análise de Sistemas
	Arquitetura de Computadores
	Atividade Extensionista I
	Atividade Extensionista II
	Banco de Dados
	Empreendedorismo
	Engenharia de Software
	Fundamentos de Sistemas de Informação
	Gestão de Projetos de Software
	Lógica de Programação e Algoritmos
	Matemática Computacional
	Programação de Computadores
	Programação Orientada a Objetos
	Sistema Gerenciador de Banco de Dados
	Sistemas Operacionais

**Pontuação:** 10 pontos.

4. Implemente uma consulta para listar o nome de todos os cursos e o nome de seus respectivos alunos. A listagem deve ser mostrada em ordem decrescente pelo nome dos cursos.

**SELECT C.nome AS Curso, A.nome AS Aluno  
FROM Curso C  
LEFT JOIN AlunoCurso AC ON C.idCurso = AC.idCurso  
LEFT JOIN Aluno A ON AC.idAluno = A.idAluno  
ORDER BY C.nome DESC, A.nome;**

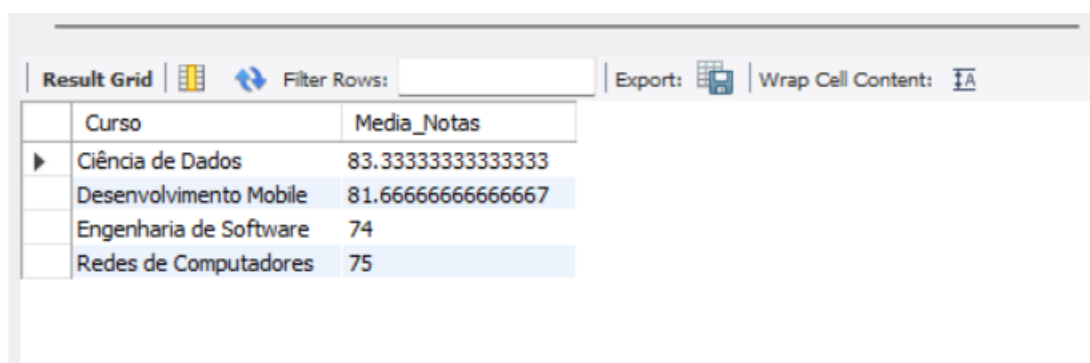


Curso	Aluno
Redes de Computadores	Nicole Amanda de Jesus
Redes de Computadores	Vitor Martins
Jogos Digitais	Beatriz Leopoldina
Jogos Digitais	João Augusto de Moura
Gestão da Tecnologia da Informação	Matheus Murilo de Souza
Gestão da Tecnologia da Informação	Miriam Miranda
Engenharia de Software	Mario Vicente
Engenharia de Software	Paula Roberta Vitorino
Engenharia da Computação	Antônio Cozer
Engenharia da Computação	Viviane Chaves Filha
Desenvolvimento Mobile	Luciano Tucolo
Desenvolvimento Mobile	Marta da Silva
Ciência de Dados	Guilherme Koeriche
Ciência de Dados	Maria Helena Mantovani
Banco de Dados	Ana Luiza de Paula
Banco de Dados	Lucas Cochuelo
Análise e Desenvolvimento de Siste...	Alice de Souza
Análise e Desenvolvimento de Siste...	Diogo Furlan
Análise e Desenvolvimento de Siste...	Marcelo Luis dos Santos

**Pontuação:** 10 pontos.

5. Implemente uma consulta para listar o nome das disciplinas e a média das notas das disciplinas em todos os cursos. Para isso, utilize o comando *group by*.

```
SELECT C.nome AS Curso, AVG(H.nota) AS Media_Notas  
FROM Curso C  
JOIN AlunoCurso AC ON C.idCurso = AC.idCurso  
JOIN Aluno A ON AC.idAluno = A.idAluno  
JOIN Historico H ON A.idAluno = H.idAluno  
GROUP BY C.nome;
```

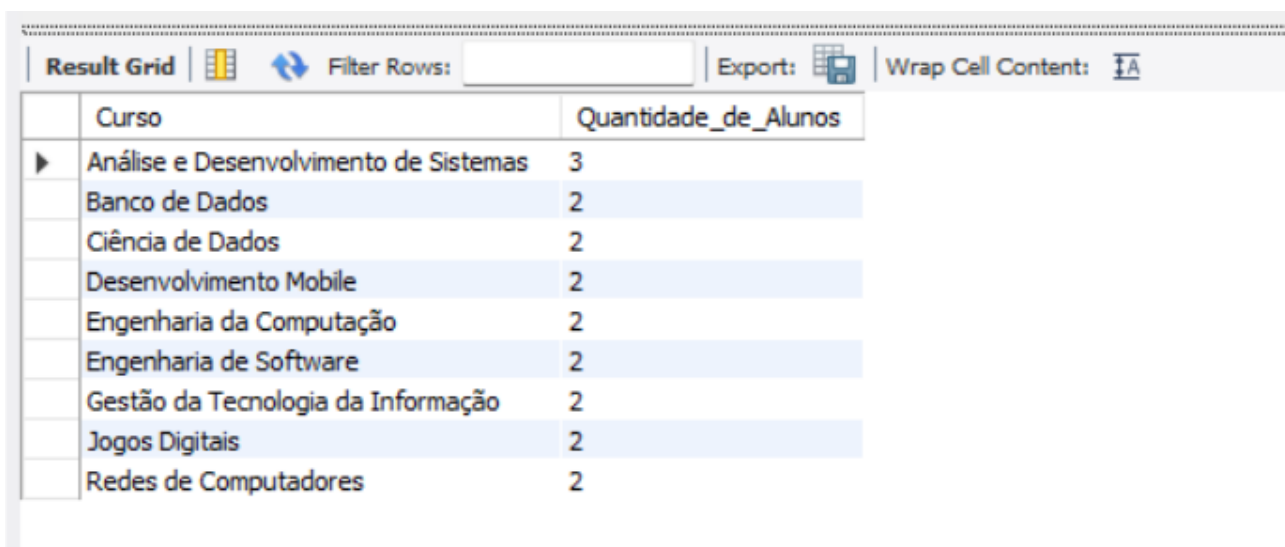


Curso	Media_Notas
Ciência de Dados	83.33333333333333
Desenvolvimento Mobile	81.66666666666667
Engenharia de Software	74
Redes de Computadores	75

**Pontuação:** 10 pontos.

6. Implemente uma consulta para listar o nome de todos os cursos e a quantidade de alunos em cada curso. Para isso, utilize os comandos *join* e *group by*.

```
SELECT C.nome AS Curso, COUNT(AC.idAluno) AS Quantidade_de_Alunos  
FROM Curso C  
LEFT JOIN AlunoCurso AC ON C.idCurso = AC.idCurso  
GROUP BY C.nome;
```



	Curso	Quantidade_de_Alunos
▶	Análise e Desenvolvimento de Sistemas	3
	Banco de Dados	2
	Ciência de Dados	2
	Desenvolvimento Mobile	2
	Engenharia da Computação	2
	Engenharia de Software	2
	Gestão da Tecnologia da Informação	2
	Jogos Digitais	2
	Redes de Computadores	2