

BANCO DE DADOS

Trabalho – Relatório

Curso:	CST ANÁLISE E DESENVOLVIMENTOS DE SISTEMAS
Aluno(a):	Rafael Terra Castilho dos Anjos
RU:	4423657

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:

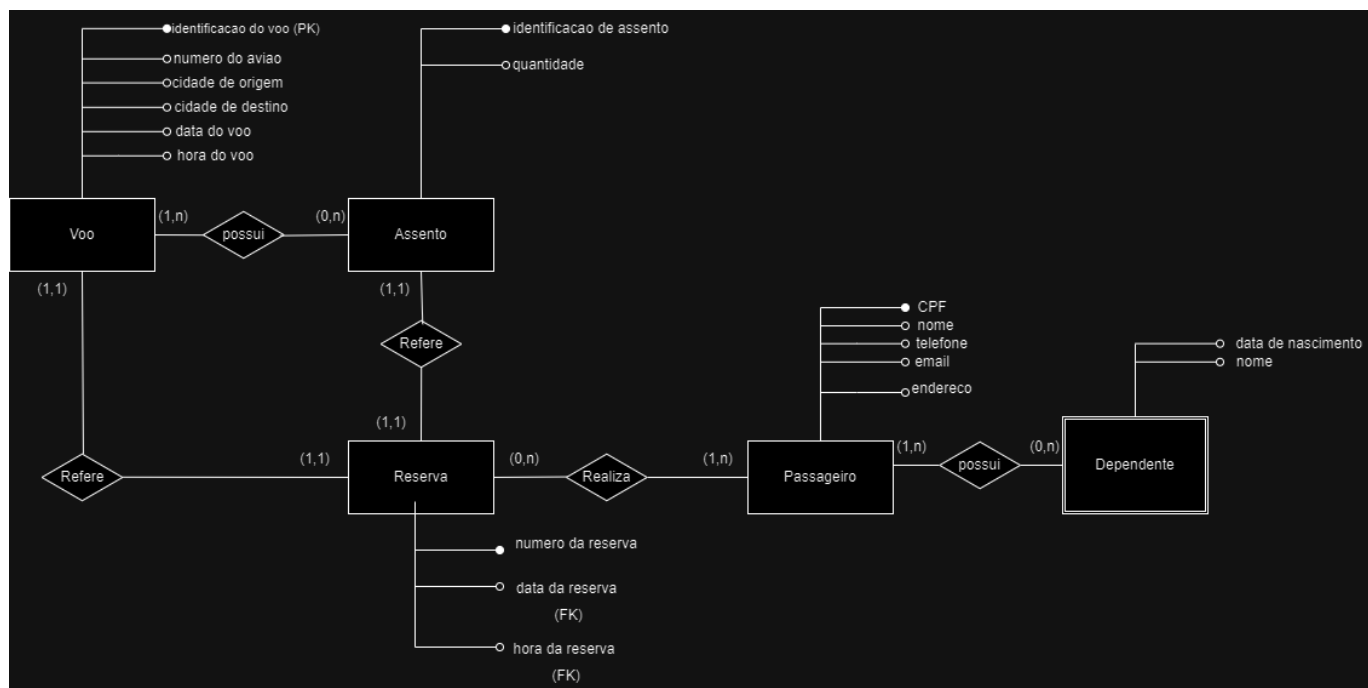
- Entidades;
- Atributos;
- Relacionamentos;
- Cardinalidades;
- Chaves primárias;
- Chaves estrangeiras.

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:

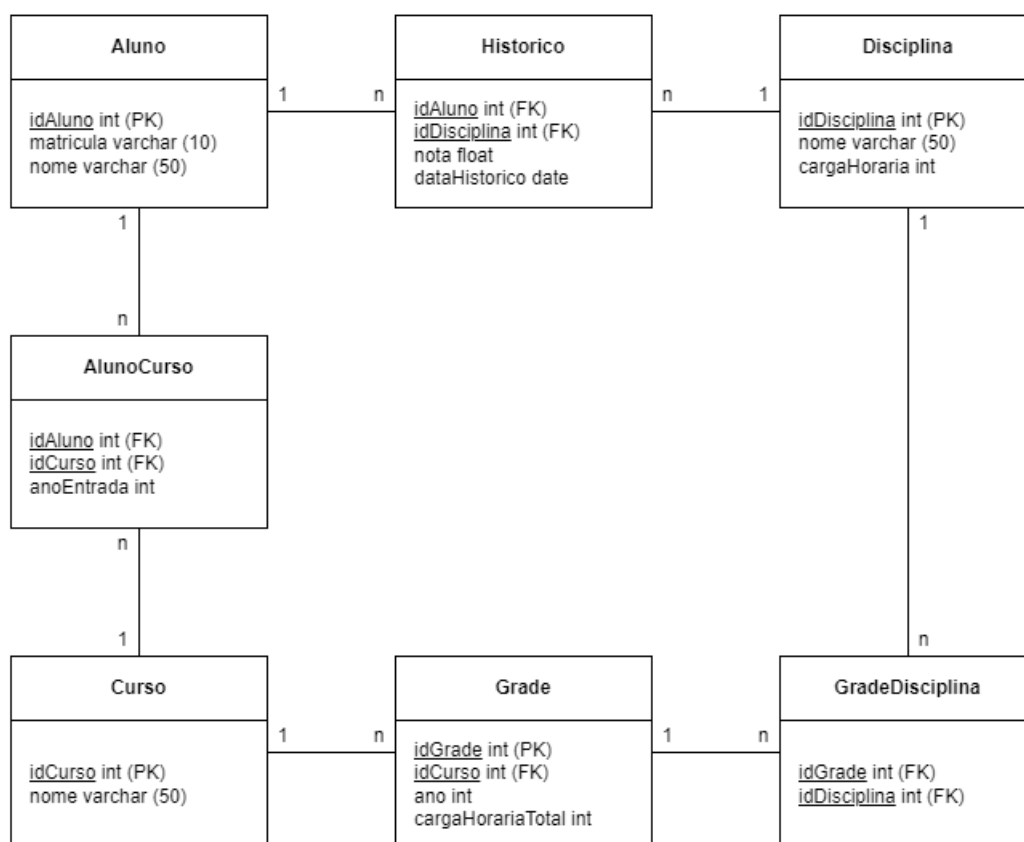
- Voo – Deverão ser armazenados os seguintes dados: identificação do voo, número do avião, cidade de origem, cidade de destino, data do voo e hora do voo;
- Assento – Deverão ser armazenados os seguintes dados: identificação do assento e quantidade;

- 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;
```

```
-- Tabela Aluno com idAluno como chave primária auto-incrementável
```

```
CREATE TABLE Aluno (  
    idAluno INT AUTO_INCREMENT PRIMARY KEY,  
    matricula VARCHAR(10) NOT NULL,  
    nome VARCHAR(100) NOT NULL  
);
```

```
-- Tabela Disciplina com idDisciplina como chave primária auto-incrementável
```

```
CREATE TABLE Disciplina (  
    idDisciplina INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(100) NOT NULL,  
    cargaHoraria INT NOT NULL  
);
```

-- Tabela Curso com idCurso como chave primária auto-incrementável

```
CREATE TABLE Curso (  
    idCurso INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(100) NOT NULL  
);
```

/* Tabela Historico com chaves primárias compostas idAluno e idDisciplina
e restrições de chave estrangeira */

```
CREATE TABLE Historico (  
    idAluno INT NOT NULL,  
    idDisciplina INT NOT NULL,  
    nota INT NOT NULL,  
    dataHistorico DATE NOT NULL,  
    PRIMARY KEY (idAluno, idDisciplina),  
    FOREIGN KEY (idAluno) REFERENCES Aluno (idAluno),  
    FOREIGN KEY (idDisciplina) REFERENCES Disciplina (idDisciplina)  
);
```

-- Tabela AlunoCurso com chaves primárias compostas idAluno e idCurso

```
CREATE TABLE AlunoCurso (  
    idAluno INT NOT NULL,  
    idCurso INT NOT NULL,  
    anoEntrada INT NOT NULL,  
    PRIMARY KEY (idAluno, idCurso)  
);
```

-- Tabela Grade com idGrade como chave primária auto-incrementável

```
CREATE TABLE Grade (  
    idGrade INT AUTO_INCREMENT PRIMARY KEY,  
    idCurso INT NOT NULL,  
    ano INT NOT NULL,  
    cargaHorariaTotal INT NOT NULL,  
    FOREIGN KEY (idCurso) REFERENCES Curso (idCurso)
```

);

/* Tabela GradeDisciplina com chaves primárias compostas idGrade e idDisciplina e restrições de chave estrangeira */

```
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)  
);
```

```
insert into Aluno (idAluno, matricula, nome) values    -- inserir valores na tabela Aluno  
( 1, 'ADS001', 'Alice de Souza'),  
( 2, 'BDS001', 'Ana Luiza de Paula'),  
( 3, 'CDS001', 'Maria Helena Mantovani'),  
( 4, 'DSM001', 'Marta da Silva'),  
( 5, 'ENC001', 'Viviane Chaves Filha'),  
( 6, 'ENS001', 'Paula Roberta Vitorino'),  
( 7, 'GTI001', 'Miriam Miranda'),  
( 8, 'JDS001', 'Beatriz Leopoldina'),  
( 9, 'RCS001', 'Nicole Amanda de Jesus'),  
(10, 'RCS002', 'Vitor Martins'),  
(11, 'JDS002', 'João Augusto de Moura'),  
(12, 'GTI002', 'Matheus Murilo de Souza'),  
(13, 'ENS002', 'Mario Vicente'),  
(14, 'ENC002', 'Antônio Cozer'),  
(15, 'DSM002', 'Luciano Tucolo'),  
(16, 'CDS002', 'Guilherme Koeriche'),  
(17, 'BDS002', 'Lucas Cochuelo'),  
(18, 'ADS002', 'Diogo Furlan'),  
(19, 'ADS003', 'Marcelo Luis dos Santos');
```

insert into Disciplina (idDisciplina, nome, cargaHoraria) values -- inserir valores na tabela Disciplina

```
( 1, 'Análise de Sistemas', 60),  
( 2, 'Arquitetura de Computadores', 60),  
( 3, 'Atividade Extensionista I', 40),  
( 4, 'Atividade Extensionista II', 40),  
( 5, 'Banco de Dados', 60),  
( 6, 'Empreendedorismo', 40),  
( 7, 'Engenharia de Software', 60),  
( 8, 'Fundamentos de Sistemas de Informação', 60),  
( 9, 'Gestão de Projetos de Software', 60),  
(10, 'Lógica de Programação e Algoritmos', 80),  
(11, 'Matemática Computacional', 40),  
(12, 'Programação de Computadores', 80),  
(13, 'Programação Orientada a Objetos', 80),  
(14, 'Sistema Gerenciador de Banco de Dados', 60),  
(15, 'Sistemas Operacionais', 60);
```

insert into Curso (idCurso, nome) values -- inserir valores na tabela Curso

```
(1, 'Análise e Desenvolvimento de Sistemas'),  
(2, 'Banco de Dados'),  
(3, 'Ciência de Dados'),  
(4, 'Desenvolvimento Mobile'),  
(5, 'Engenharia da Computação'),  
(6, 'Engenharia de Software'),  
(7, 'Gestão da Tecnologia da Informação'),  
(8, 'Jogos Digitais'),  
(9, 'Redes de Computadores');
```

insert into Historico (idAluno, idDisciplina, nota, dataHistorico) values -- inserir valores na tabela historico

```
( 3, 1, 90, '2022-12-09'),  
( 3, 3, 75, '2022-12-09'),  
( 3, 5, 85, '2022-12-09'),
```

```
( 9, 1, 80, '2022-12-16'),  
( 9, 9, 75, '2022-12-16'),  
( 9, 11, 70, '2022-12-16'),  
(13, 12, 70, '2022-12-09'),  
(13, 13, 70, '2022-12-09'),  
(13, 14, 82, '2022-12-09'),  
(15, 2, 76, '2022-12-16'),  
(15, 4, 80, '2022-12-16'),  
(15, 6, 89, '2022-12-16');
```

insert into AlunoCurso (idAluno, idCurso, anoEntrada) values -- inserir valores na
tabela AlunoCurso

```
( 1, 1, 2023),  
( 2, 2, 2023),  
( 3, 3, 2022),  
( 4, 4, 2023),  
( 5, 5, 2023),  
( 6, 6, 2023),  
( 7, 7, 2023),  
( 8, 8, 2023),  
( 9, 9, 2022),  
(10, 9, 2023),  
(11, 8, 2023),  
(12, 7, 2023),  
(13, 6, 2022),  
(14, 5, 2023),  
(15, 4, 2022),  
(16, 3, 2023),  
(17, 2, 2023),  
(18, 1, 2023),  
(19, 1, 2023);
```

insert into Grade (idGrade, idCurso, ano, cargaHorariaTotal) values -- inserir valores
na tabela grade

(1, 1, 2021, 880),
(2, 2, 2022, 880),
(3, 3, 2022, 880),
(4, 4, 2022, 880),
(5, 5, 2019, 880),
(6, 6, 2022, 880),
(7, 7, 2022, 880),
(8, 8, 2022, 880),
(9, 9, 2019, 880),
(10, 1, 2023, 880),
(11, 5, 2023, 880),
(12, 9, 2023, 880);

insert into GradeDisciplina (idGrade, idDisciplina) values -- inserir valores na tabela GradeDisciplina

(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11), (1, 12), (1, 13), (1, 14), (1, 15),
(2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (2, 12), (2, 13), (2, 14), (2, 15),
(3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (3, 11), (3, 12), (3, 13), (3, 14), (3, 15),
(4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (4, 10), (4, 11), (4, 12), (4, 13), (4, 14), (4, 15),
(5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (5, 7), (5, 8), (5, 9), (5, 10), (5, 11), (5, 12), (5, 13), (5, 14), (5, 15),
(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6), (6, 7), (6, 8), (6, 9), (6, 10), (6, 11), (6, 12), (6, 13), (6, 14), (6, 15),
(7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8), (7, 9), (7, 10), (7, 11), (7, 12), (7, 13), (7, 14), (7, 15),
(8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 8), (8, 9), (8, 10), (8, 11), (8, 12), (8, 13), (8, 14), (8, 15),
(9, 1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8), (9, 9), (9, 10), (9, 11), (9, 12), (9, 13), (9, 14), (9, 15),

(10, 1), (10, 2), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7), (10, 8), (10, 9), (10, 10), (10, 11), (10, 12), (10, 13), (10, 14), (10, 15),
(11, 1), (11, 2), (11, 3), (11, 4), (11, 5), (11, 6), (11, 7), (11, 8), (11, 9), (11, 10), (11, 11), (11, 12), (11, 13), (11, 14), (11, 15),
(12, 1), (12, 2), (12, 3), (12, 4), (12, 5), (12, 6), (12, 7), (12, 8), (12, 9), (12, 10), (12, 11), (12, 12), (12, 13), (12, 14), (12, 15);

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

```
SELECT nome  
FROM Disciplina;
```

```
SELECT c.nome AS NomeDoCurso, a.nome AS NomeDoAluno  
FROM Curso c  
INNER JOIN AlunoCurso ac ON c.idCurso = ac.idCurso  
INNER JOIN Aluno a ON ac.idAluno = a.idAluno  
ORDER BY c.nome DESC;
```

```
SELECT d.nome AS NomeDaDisciplina, AVG(h.nota) AS MediaDeNotas  
FROM Disciplina d  
LEFT JOIN Historico h ON d.idDisciplina = h.idDisciplina  
GROUP BY d.nome;
```

```
SELECT c.nome AS NomeDoCurso, COUNT(ac.idAluno) AS QuantidadeDeAlunos  
FROM Curso c  
LEFT JOIN AlunoCurso ac ON c.idCurso = ac.idCurso  
GROUP BY c.nome;
```

Pontuação: 10 pontos.

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

```
SELECT COUNT(*) AS QuantidadeDeCursos
```

FROM Curso;

	QuantidadeDeCursos
▶	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 NomeDoCurso, a.nome AS NomeDoAluno
FROM Curso c
INNER JOIN AlunoCurso ac ON c.idCurso = ac.idCurso
INNER JOIN Aluno a ON ac.idAluno = a.idAluno
ORDER BY c.nome DESC;
```

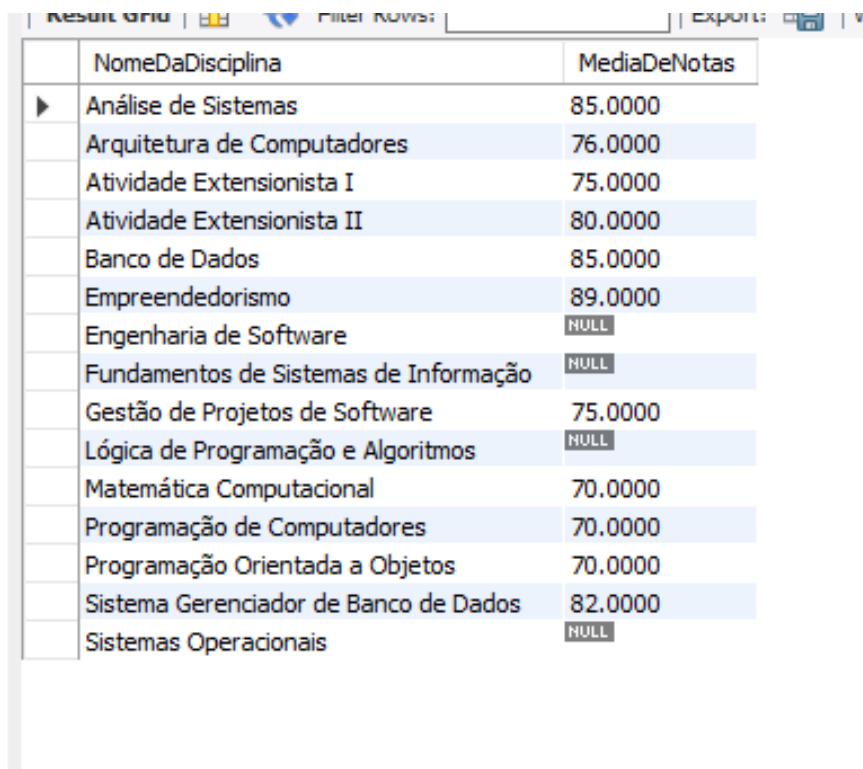
NomeDoCurso	NomeDoAluno
Redes de Computadores	Vitor Martins
Redes de Computadores	Nicole Amanda de Jesus
Jogos Digitais	João Augusto de Moura
Jogos Digitais	Beatriz Leopoldina
Gestão da Tecnologia da Informação	Miriam Miranda
Gestão da Tecnologia da Informação	Matheus Murilo de Souza
Engenharia de Software	Paula Roberta Vitorino
Engenharia de Software	Mario Vicente
Engenharia da Computação	Viviane Chaves Filha
Engenharia da Computação	Antônio Cozer
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...	Marcelo Luis dos Santos
Análise e Desenvolvimento de Siste...	Diogo Furlan
Análise e Desenvolvimento de Siste...	Alice de Souza

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 d.nome AS NomeDaDisciplina, AVG(h.nota) AS MediaDeNotas
FROM Disciplina d
LEFT JOIN Historico h ON d.idDisciplina = h.idDisciplina
```

GROUP BY d.nome;



NomeDaDisciplina	MediaDeNotas
Análise de Sistemas	85.0000
Arquitetura de Computadores	76.0000
Atividade Extensionista I	75.0000
Atividade Extensionista II	80.0000
Banco de Dados	85.0000
Empreendedorismo	89.0000
Engenharia de Software	NULL
Fundamentos de Sistemas de Informação	NULL
Gestão de Projetos de Software	75.0000
Lógica de Programação e Algoritmos	NULL
Matemática Computacional	70.0000
Programação de Computadores	70.0000
Programação Orientada a Objetos	70.0000
Sistema Gerenciador de Banco de Dados	82.0000
Sistemas Operacionais	NULL

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 NomeDoCurso, COUNT(ac.idAluno) AS QuantidadeDeAlunos  
FROM Curso c  
LEFT JOIN AlunoCurso ac ON c.idCurso = ac.idCurso  
GROUP BY c.nome;
```

Result Grid			Filter Rows:	Export:	Wrap C
	NomeDoCurso	QuantidadeDeAlunos			
▶	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			