BANCO DE DADOS
Trabalho – Relatório

Curso:	ANALISE E DESENVOLVIMENTO DE SISTEMAS
Aluno(a):	FELIPE DE MELO DOS SANTOS REIZ
RU:	4302932

- ### 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.

Cole o Modelo Entidade-Relacionamento (MER) aqui.

Entidades:

Voo:
Identificação do Voo (Chave Primária)
Número do Avião
Cidade de Origem
Cidade de Destino
Data do Voo
Hora do Voo

Assento:
Identificação do Assento (Chave Primária)
Quantidade

Passageiro:
CPF (Chave Primária)

Nome
Telefone
E-mail
Endereço (Rua, Número, Complemento, Bairro, CEP, Cidade, Estado)

Dependente:
Nome
Data de Nascimento

Reserva de Assento:
Identificação do Voo (Chave Estrangeira)
Identificação do Assento (Chave Estrangeira)
CPF do Passageiro (Chave Estrangeira)
Data da Reserva
Hora da Reserva

Relacionamentos:

Um Voo pode ter zero ou vários Assentos.
Um Assento pertence a um Voo.
Um Passageiro pode fazer zero ou várias Reservas de Assentos.
Uma Reserva de Assento pertence a um Passageiro.
Um Passageiro pode ter zero ou vários Dependentes.
Um Dependente pertence a um Passageiro.

Cardinalidades:

Voo (1) - (0,N) Assento
Passageiro (1) - (0,N) Reserva de Assento
Passageiro (1) - (0,N) Dependente

Chaves Primárias:

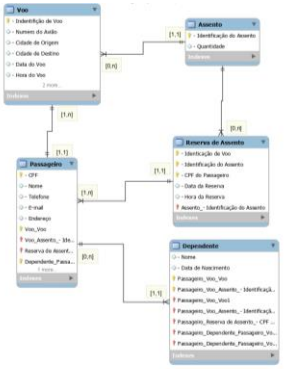
Voo:
Identificação do Voo

Assento:
Identificação do Assento

Passageiro: CPF

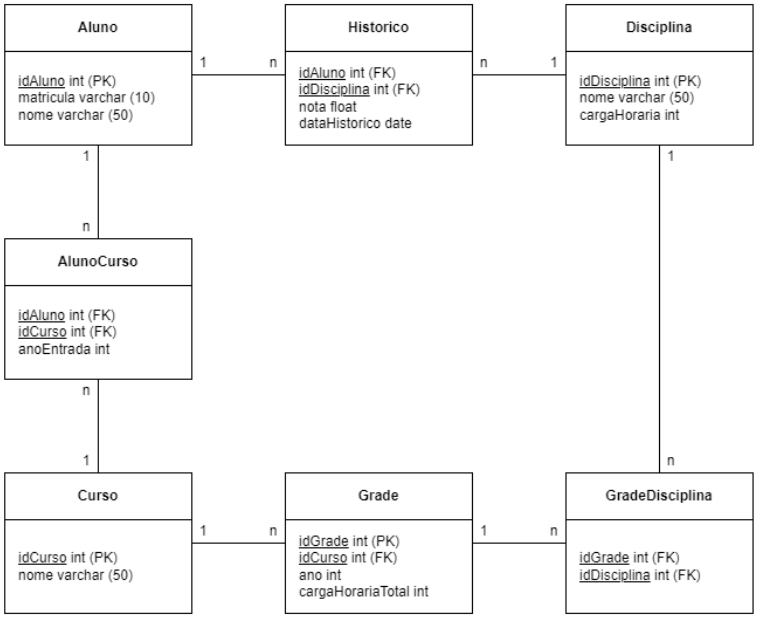
Reserva de Assento:
Identificação do Voo, Identificação do Assento e CPF do Passageiro

Dependentes:
Dependentes não tem chaves primarias



- **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.

- 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*).

Cole o código aqui.

```
CREATE DATABASE IF NOT EXISTS Faculdade;
USE Faculdade;

CREATE TABLE IF NOT EXISTS Aluno (
    idAluno INT,
    matricula VARCHAR(10),
    nome VARCHAR(50),
    PRIMARY KEY (idAluno)
);

CREATE TABLE IF NOT EXISTS Disciplina (
    idDisciplina INT,
    nome VARCHAR(50),
    cargaHoraria INT,
    PRIMARY KEY (idDisciplina)
);

CREATE TABLE IF NOT EXISTS Curso (
    idCurso INT,
    nome VARCHAR(50),
    PRIMARY KEY (idCurso)
);

CREATE TABLE IF NOT EXISTS AlunoCurso (
    idAluno INT,
    idCurso INT,
    anoEntrada INT,
    PRIMARY KEY (idAluno, idCurso),
```

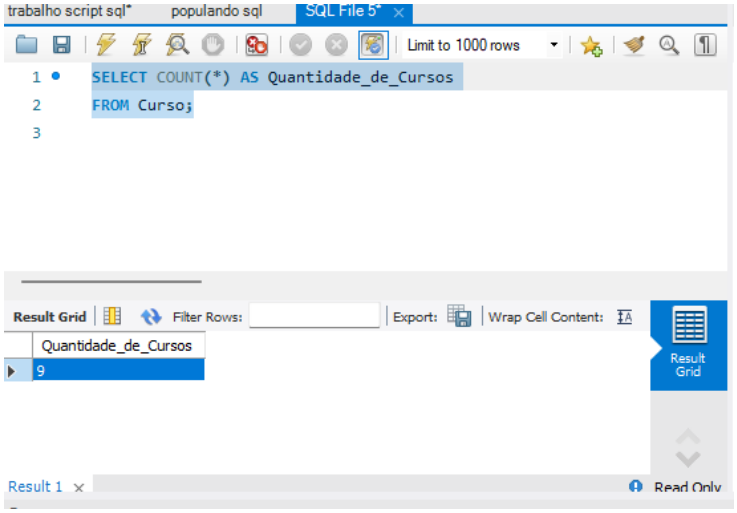
```
    FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),
    FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)
);

CREATE TABLE IF NOT EXISTS Grade (
    idGrade INT,
    idCurso INT,
    ano INT,
    cargaHorariaTotal INT,
    PRIMARY KEY (idGrade),
    FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)
);

CREATE TABLE IF NOT EXISTS GradeDisciplina (
    idGrade INT,
    idDisciplina INT,
    PRIMARY KEY (idGrade, idDisciplina),
    FOREIGN KEY (idGrade) REFERENCES Grade(idGrade),
    FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)
);

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

Pontuação: 10 pontos.

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

Cole o código e o *print* resultante da consulta aqui.

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

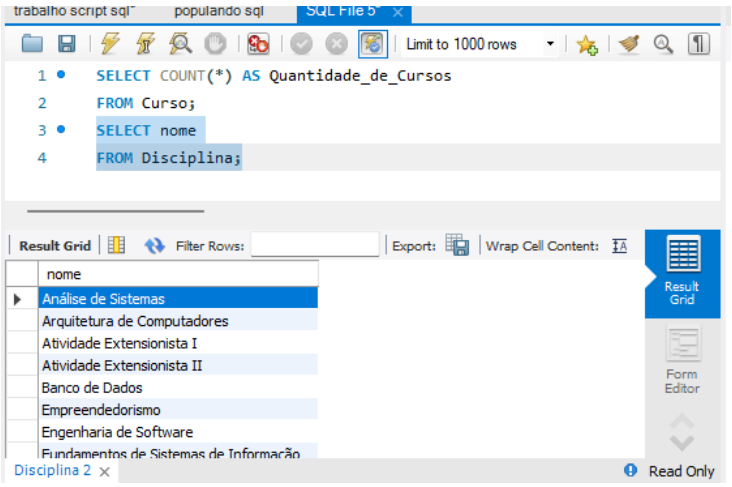


Pontuação: 10 pontos.

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

Cole o código e o *print* resultante da consulta aqui.

```
SELECT nome
FROM Disciplina;
```



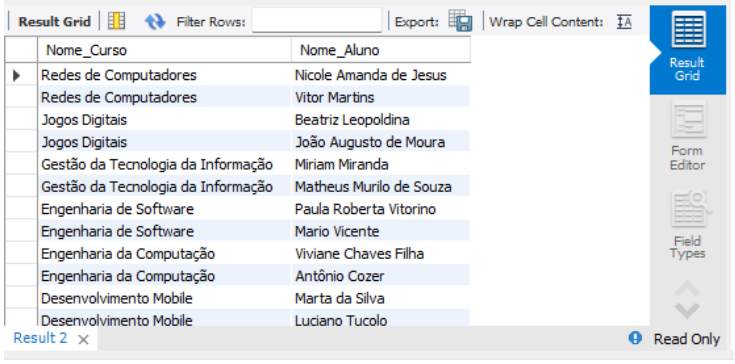
Pontuação: 10 pontos.

- 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.

Cole o código e o *print* resultante da consulta aqui.

```
SELECT Curso.nome AS Nome_Curso, Aluno.nome AS Nome_Aluno
FROM Curso
LEFT JOIN AlunoCurso ON Curso.idCurso = AlunoCurso.idCurso
LEFT JOIN Aluno ON AlunoCurso.idAluno = Aluno.idAluno
ORDER BY Curso.nome DESC;
```

```
1  SELECT Curso.nome AS Nome_Curso, Aluno.nome AS Nome_Aluno
2  FROM Curso
3  LEFT JOIN AlunoCurso ON Curso.idCurso = AlunoCurso.idCurso
4  LEFT JOIN Aluno ON AlunoCurso.idAluno = Aluno.idAluno
5  ORDER BY Curso.nome DESC;
```



Nome_Curso	Nome_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	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	Marta da Silva
Desenvolvimento Mobile	Luciano Tucolo

Pontuação: 10 pontos.

- 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*.

Cole o código e o *print* resultante da consulta aqui.

```
SELECT Disciplina.nome AS Nome_Disciplina, AVG(Historico.nota) AS Media_Notas
FROM Disciplina
LEFT JOIN Historico ON Disciplina.idDisciplina = Historico.idDisciplina
GROUP BY Disciplina.nome;
```

```
SELECT Disciplina.nome AS Nome_Disciplina, AVG(Historico.nota) AS Media_Notas
FROM Disciplina
LEFT JOIN Historico ON Disciplina.idDisciplina = Historico.idDisciplina
GROUP BY Disciplina.nome;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Nome_Disciplina	Media_Notas
Análise de Sistemas	85
Arquitetura de Computadores	76
Atividade Extensionista I	75
Atividade Extensionista II	80
Banco de Dados	85
Empreendedorismo	89
Engenharia de Software	NULL
Fundamentos de Sistemas de Informação	NULL
Gestão de Projetos de Software	75
Lógica de Programação e Algoritmos	NULL
Matemática Computacional	70
Programação de Computadores	70
Programação Orientada a Objetos	70
Sistema Gerenciador de Banco de Dados	82
Sistemas Operacionais	NULL

Result Grid | Form Editor | Field Types | Query Stats

Result 1 | Read Only

Comentado [FM1]: não entendi e não consegui encontra o erro de algumas media terem dado "NULL"

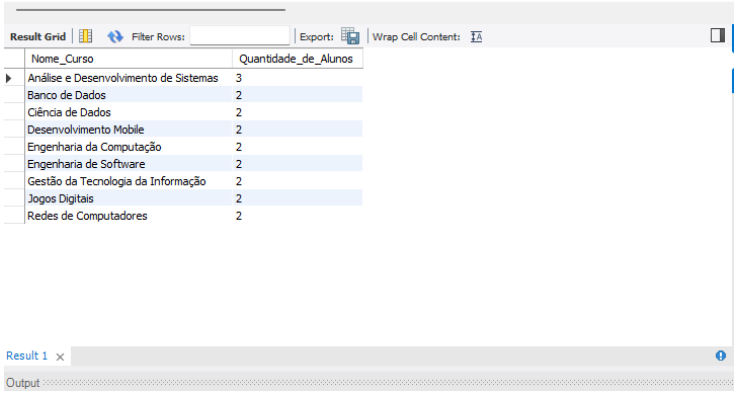
Pontuação: 10 pontos.

- 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*.

Cole o código e o *print* resultante da consulta aqui.

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

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



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Result 1

Output