**BANCO DE DADOS**

**Trabalho – Relatório**

|  |  |
| --- | --- |
| **Curso:** | CST ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - DISTÂNCIA |
| **Aluno(a):** | Matheus Belarmino Pignata |
| **RU:** | 4525875 |

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.

**Cole o Modelo Entidade-Relacionamento (MER) aqui.**

1. **2ª Etapa – Implementação**

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

Diagrama

Descrição gerada automaticamente

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 PRIMARY KEY NOT NULL,**

**matricula VARCHAR(10) NOT NULL,**

**nome VARCHAR(50) NOT NULL**

**);**

**CREATE TABLE Disciplina (**

**idDisciplina INT PRIMARY KEY NOT NULL,**

**nome VARCHAR(50) NOT NULL,**

**cargaHoraria INT NOT NULL**

**);**

**CREATE TABLE Curso (**

**idCurso INT PRIMARY KEY NOT NULL,**

**nome VARCHAR(50) NOT NULL**

**);**

**CREATE TABLE Historico (**

**idHistorico INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,**

**idAluno INT NOT NULL,**

**idDisciplina INT NOT NULL,**

**nota FLOAT NOT NULL,**

**dataHistorico DATE NOT NULL,**

**FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),**

**FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)**

**);**

**CREATE TABLE AlunoCurso (**

**idAlunoCurso INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,**

**idAluno INT NOT NULL,**

**idCurso INT NOT NULL,**

**anoEntrada INT NOT NULL,**

**FOREIGN KEY (idAluno) REFERENCES Aluno(idAluno),**

**FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)**

**);**

**CREATE TABLE Grade (**

**idGrade INT PRIMARY KEY NOT NULL,**

**idCurso INT NOT NULL,**

**ano INT NOT NULL,**

**cargaHorariaTotal INT NOT NULL,**

**FOREIGN KEY (idCurso) REFERENCES Curso(idCurso)**

**);**

**CREATE TABLE GradeDisciplina (**

**idGradeDisciplina INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,**

**idGrade INT NOT NULL,**

**idDisciplina INT NOT NULL,**

**FOREIGN KEY (idGrade) REFERENCES Grade(idGrade),**

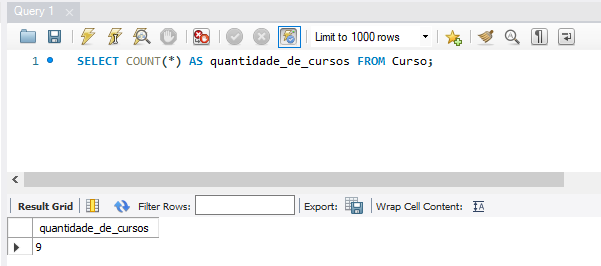
**FOREIGN KEY (idDisciplina) REFERENCES Disciplina(idDisciplina)**

**);**

**Pontuação:** 10 pontos.

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

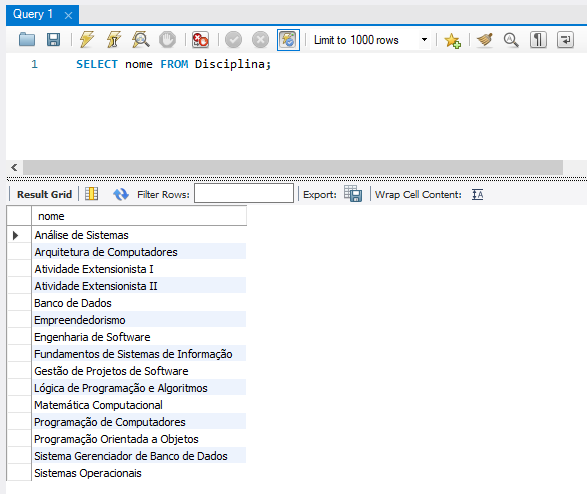
**SELECT COUNT(\*) AS quantidade\_de\_cursos FROM Curso;**

****

**Pontuação:** 10 pontos.

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

**SELECT nome FROM Disciplina;**



**Pontuação:** 10 pontos.

1. 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.

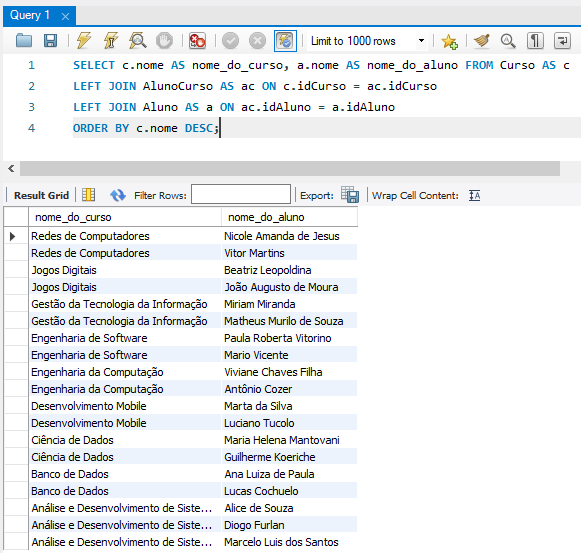
**SELECT c.nome AS nome\_do\_curso, a.nome AS nome\_do\_aluno**

**FROM Curso AS c**

**LEFT JOIN AlunoCurso AS ac ON c.idCurso = ac.idCurso**

**LEFT JOIN Aluno AS a ON ac.idAluno = a.idAluno**

**ORDER BY c.nome DESC;**



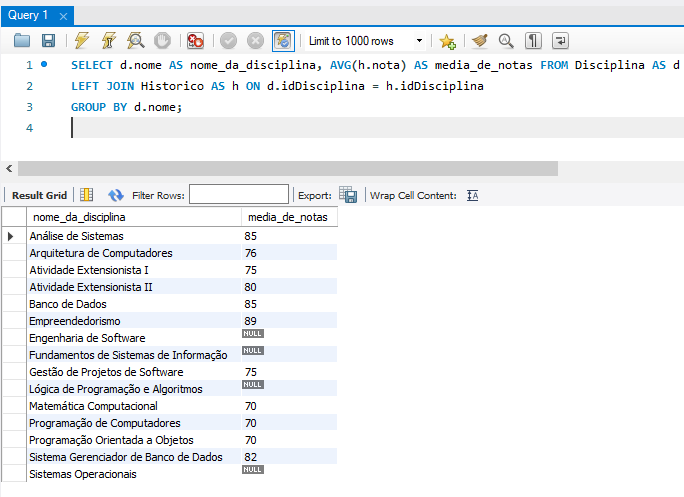
**Pontuação:** 10 pontos.

1. 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 nome\_da\_disciplina, AVG(h.nota) AS media\_de\_notas FROM Disciplina AS d**

**LEFT JOIN Historico AS h ON d.idDisciplina = h.idDisciplina**

**GROUP BY d.nome;**



**Pontuação:** 10 pontos.

1. 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 nome\_do\_curso, COUNT(ac.idAluno) AS quantidade\_de\_alunos FROM Curso AS c**

**LEFT JOIN AlunoCurso AS ac ON c.idCurso = ac.idCurso**

**GROUP BY c.nome;**

