Departamento de Ciência da Computação
UFLA - Universidade Federal de Lavras
GCC214 - Introdução a Sistemas de Banco de Dados
Prof. Denilson Alves Pereira
Trabalho Prático – Etapa/3/3

Grupo 6 - Integrantes - 10A:

- ➔ **David de Jesus Costa - 202020086**
- ➔ **Diego de Souza Marques - 202111037**
- ➔ **Mauricio Martins Damasceno - 202020884**
- ➔ **Igor Cunha Ferreira - 201720493**

Plataforma de Cursos Alura

Descrição:

Modelagem de dados para uma plataforma de streaming de cursos, Alura, voltados para a área de tecnologia da informação (TI). O funcionamento da plataforma é da seguinte maneira, o aluno cadastrado na plataforma "Alura", possui o direito de matricular-se em quantos cursos quiser, licenciado por um ou mais professores, nos quais o aluno pode ou não obter um certificado de conclusão, dependendo do seu progresso no curso. Além disso, funcionários podem participar de eventos, de modo a obter horas de participação solicitadas pelas empresas. É importante armazenar dos alunos, seu número de identificação, nome e sexo. Ademais, também é importante armazenar os números individuais de cada curso para saber quais cursos a plataforma possui e quais alunos fizeram tais cursos. Dos eventos, é importante armazenar o nome, o número individual de cada evento, e o período total de cada evento. Também, é importante saber as horas de acesso de cada funcionário em um evento.

Tipo de Entidade	Professor		
Descrição	Conjunto de professores que licencia cursos em uma plataforma de ensino Alura		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
idProfessor	Cadastro de individual	Int	N

	de cada professor		
nomeProf	Nome completo do professor	Texto(70)	N
especialidade	Gradução, cursos, formação	Texto(400)	N
salário	Salário mensal	Real(8,2) positivo	N
sexo	Sexo (masculino,femino)	Text(1) M - Masculino F- Feminino	N

Tipo de Entidade	Aluno		
Descrição	conjuntos de alunos matriculados na plataforma de ensino Alura		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
idAluno	Número individual do aluno	Int	N
nomeAluno	Nome do Aluno	Texto(70)	N
sexo	Sexo (masculino ou feminino)	Text(1) M - Masculino F - Feminino	N

Tipo de Entidade	Empresa		
Descrição	Conjuntos de empresas que matricularam seus funcionários na plataforma de ensino Alura		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
idEmpresa	Número individual de cada empresa	Texto(9)	N

nomeEmpresa	Nome da empresa	Texto(30)	N
*qtd_funcionário	Quantidades de funcionários que serão matriculados na plataforma. É obtido atraves a quantidade de funcionarios vinculados a uma empresa	Inteiro positivo	N

Tipo de Entidade	Curso		
Descrição	Conjunto de cursos presentes na plataforma Alura		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
nomeCurso	Nome do curso	Texto(70)	N
idCurso	Identificação do curso	Int	N
plano	Plano contratado	Texto(70)	N

Tipo de Entidade	Matricula		
Descrição	Conjunto de matrículas realizadas por alunos em um curso		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
progressoCurso	Progresso do aluno em um curso	Inteiro positivo (2) Formato: dd%	S
dataInicio	Data de início no curso	Data Formato: dd/dd/ddd	N
*certificado	Certificado de conclusão do curso. É obtido através do progresso do curso de no mínimo 60%	Texto(1) S - Sim N - Não	N

Tipo de Entidade	Evento		
Descrição	Conjuntos de eventos imersivos desenvolvidos por professores		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
idEvento	Código individual do evento	Int d-ddddddd	N
nomeEvento	Nome do evento	Texto(20)	N
dataInicio	Data de inicio do Evento	Data Formato: dd/dd/ddd	N
dataFim	Data final do Evento	Data Formato: dd/dd/ddd	N
carga_horaria	Indica a carga horária total do evento	Inteiro positivo(4) Formato: dd:dd	N

Tipo de Entidade	Funcionário		
Descrição	Conjunto de funcionários que trabalham em uma empresa		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
idFuncionario	número individual	Int	N
nomeFuncionário	Nome do Funcionário	Texto(20)	N
cargoFuncionário	Cargo do funcionário dentro da empresa	Texto(20)	N
*hora_part_evento	Quantidade de horas total de participação em eventos, é obtido com. É obtido através da soma da diferença do atributo (Hora_Final-Hora_inicial) respectivo de cada dia da tabela participa.	Inteiro positivo(4) Formato: dd:dd	S

Tipo de Relacionamento	licencia		
Descrição	Indica os professores que leciona um curso		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

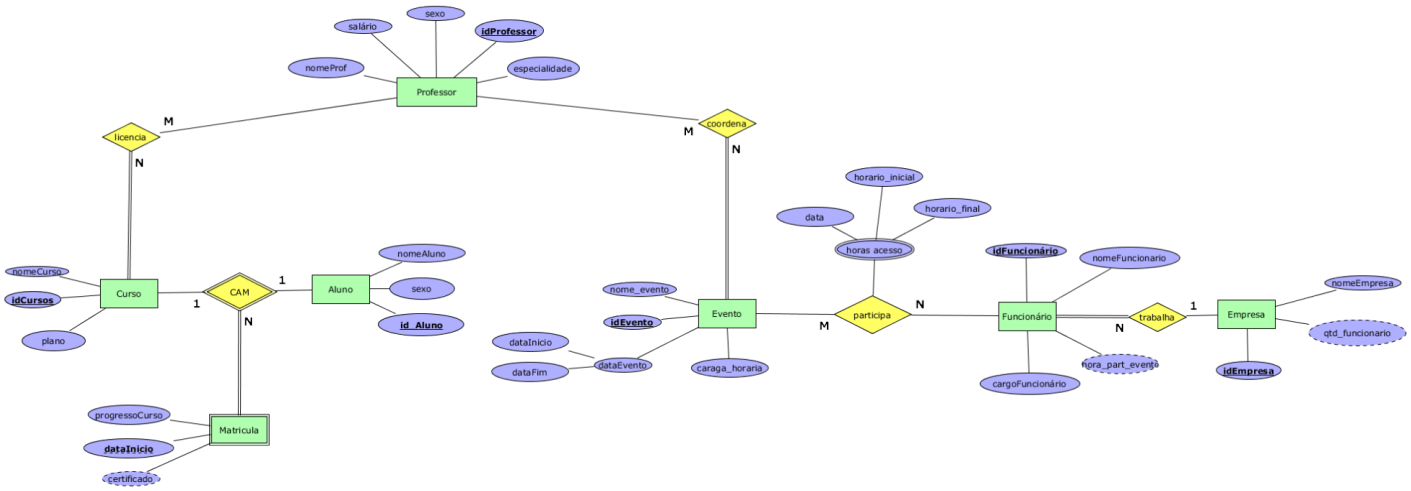
Tipo de Relacionamento	trabalha		
Descrição	Indica a empresa que os funcionários trabalham		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

Tipo de Relacionamento	coordena		
Descrição	Indica os professore(s) que coordenam um evento		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

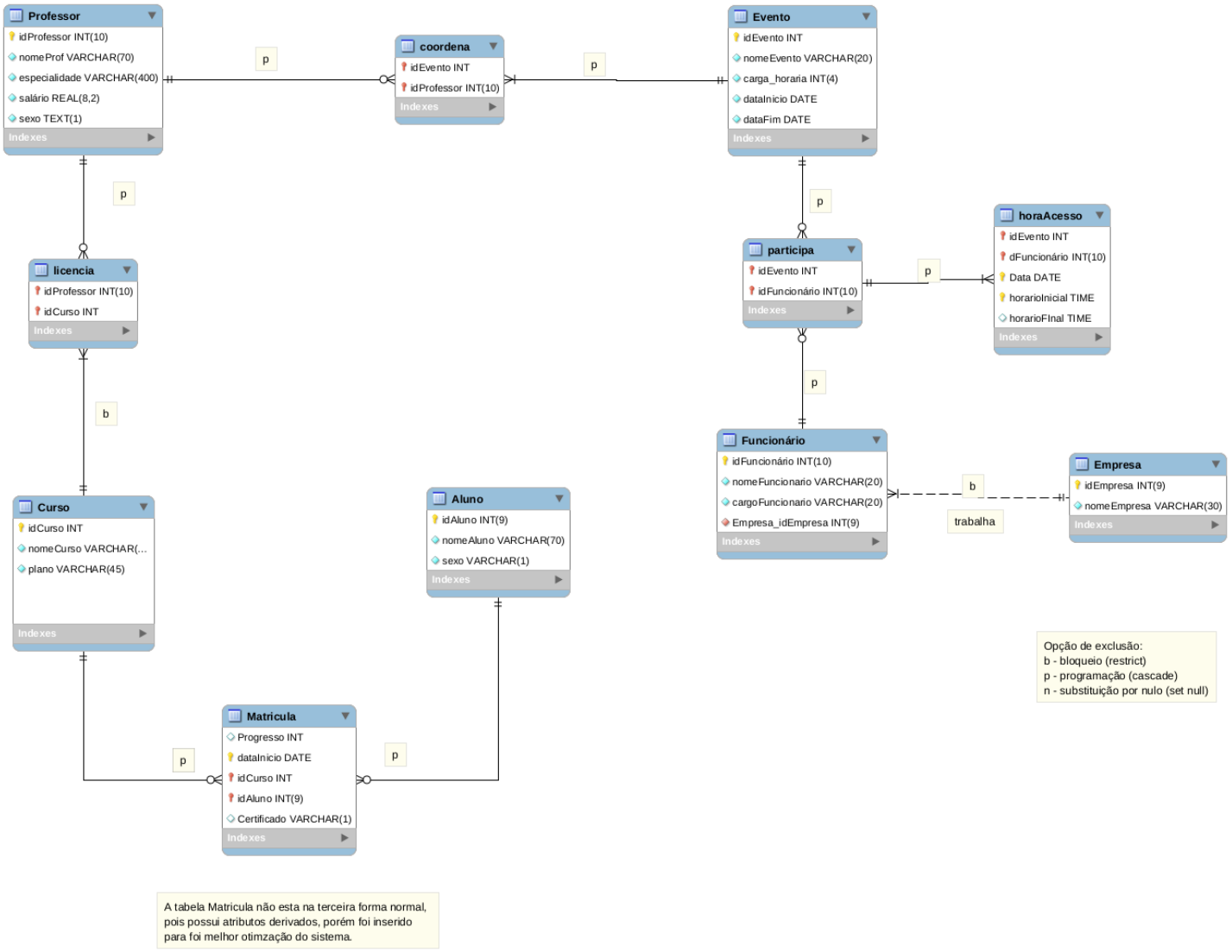
Tipo de Relacionamento	CAM		
Descrição	Indica o relacionamento ternário entre curso, aluno e matrícula, ao qual o aluno pode se matricular em um determinado curso mais de uma vez		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

Tipo de Relacionamento	participa
Descrição	Indica os eventos que os funcionários participaram

Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Horario_Inicial	Hora inicial de acesso ao evento	Inteiro positivo(4) Formato: dd:dd	S
Data	Data inicial de acesso ao evento	Data: dd/dd/dddd	**S**
Horario_Final	Horário final de acesso ao evento	Inteiro positivo(4) Formato: dd:dd	**S**



Modelo Relacional



Dicionário de Dados Relacional

Tipo de Entidade	Professor
Descrição	Professores da plataforma de ensino Alura que ministram cursos e eventos.
Atributos	
Nome	**Descrição**
idProfessor	Código identificador de professor. Auto incrementável.
nomeProf	Nome professor
especialidade	Formação acadêmica do professor
salário	Salário do professor
sexo	Sexo(masculino, feminino)

Tipo de Entidade	Evento
Descrição	Eventos coordenados por professores
Atributos	
Nome	**Descrição**
idEvento	Código de identificação do evento. Auto incrementável
nomeEvento	Nome do evento
carga_horaria	Tempo de duração do evento
dataInicio	Data do início do evento
dataFim	Data do fim do evento

Tipo de Entidade	Coordena
Descrição	Relação entre professor e evento
Atributos	
Nome	**Descrição**
idEvento	Referência ao código identificador do evento.
idProfessor	Referência ao código identificador de professor.

Tipo de Entidade	Funcionario
Descrição	Funcionário de uma empresa que participam de um evento
Atributos	
Nome	**Descrição**
idFuncionario	Código identificador de funcionário. Auto Incrementável.
nomeFuncionario	Nome do funcionário
cargoFuncionario	Função/ cargo que o funcionário exerce na empresa
idEmpresa	Referência ao identificador da empresa em que o funcionário trabalha.

Tipo de Entidade	horaAcesso
Descrição	Registrar o tempo de acesso ao evento de um funcionário
Atributos	
Nome	**Descrição**
idEvento	Referência ao identificador de evento
idFuncionario	Referência ao identificador de funcionário
Data	Data de acesso ao evento
horarioInicial	Horário de início do acesso
horarioFim	Horário final do acesso

Tipo de Entidade	Empresa
Descrição	Empresa matriculada na plataforma de ensino Alura
Atributos	
Nome	**Descrição**
idEmpresa	Código identificador da empresa. Auto incrementável
nomeEmpresa	Nome da empresa

Tipo de Entidade	Curso
Descrição	Curso lecionado pelos professores
Atributos	
Nome	**Descrição**
idCurso	Código identificador do curso. Auto incrementável
nomeCurso	Nome do curso
plano	Plano contratado pelo aluno que disponibiliza acesso aos cursos da plataforma.

Tipo de Entidade	Licencia
Descrição	Registra os professores e os cursos licenciados por eles
Atributos	
Nome	**Descrição**
idCurso	Referência ao código identificador do curso
idProfessor	Referência ao código identificador do professor

Tipo de Entidade	Aluno
Descrição	Aluno que se matriculam em cursos
Atributos	
Nome	**Descrição**
idAluno	Código identificador do aluno.Auto incrimentável
nomeAluno	Nome do aluno
sexo	Sexo(masculino, feminino)

Tipo de Entidade	Matricula
Descrição	Registra as matrículas realizadas pelos alunos em um curso
Atributos	
Nome	**Descrição**
progresso	Percentual concluído do curso
dataInicio	Data da realização da matrícula
idCurso	Referência ao código identificador do curso
idAluno	Referência ao código identificador do aluno
Certificado	Certificado de aprovação em um curso (S/N)

Tipo de Entidade	Participa
Descrição	Registra os funcionários que participaram de um evento
Atributos	
Nome	**Descrição**
idEvento	Referência ao código identificador do evento
idFuncionario	Referência ao código identificador do funcionário

SGBD Relacional

a)

```
-- Criação e utilização do SCHEMA.
CREATE SCHEMA aluraDb;
USE aluraDb;

-- Criação das tabelas e restrições
-- TABELA PROFESSOR
CREATE TABLE IF NOT EXISTS Professor (
      idProfessor INT NOT NULL AUTO_INCREMENT,
      nomeProf VARCHAR(70) NOT NULL,
      especialidade VARCHAR(400) NOT NULL,
      salario REAL NOT NULL,
      sexo TEXT(1) NOT NULL CHECK (sexo = 'M'or sexo = 'F'),
     -- Define idProfessor como chave primária
      PRIMARY KEY (idProfessor));

-- TABELA ALUNO
CREATE TABLE IF NOT EXISTS Aluno (
      idAluno INT NOT NULL AUTO_INCREMENT,
      nomeAluno VARCHAR(70) NOT NULL,
      sexo VARCHAR(1) NOT NULL CHECK (sexo = 'M'or sexo = 'F'),
     -- Define Aluno como chave primária
      PRIMARY KEY (idAluno));

-- TABELA EMPRESA
CREATE TABLE IF NOT EXISTS Empresa (
      idEmpresa INT NOT NULL AUTO_INCREMENT,
```

```
      nomeEmpresa VARCHAR(30) NOT NULL,
    -- Define idEmpresa como chave primária e nomeEmpresa como chave secundária
      PRIMARY KEY (idEmpresa),
      UNIQUE INDEX nomeEmpresa_UNIQUE (nomeEmpresa ASC) VISIBLE);

-- TABELA CURSO
CREATE TABLE IF NOT EXISTS Curso (
      idCurso INT NOT NULL AUTO_INCREMENT,
      nomeCurso VARCHAR(70) NOT NULL,
      plano VARCHAR(45) NOT NULL,
    -- Define idCurso como chave primária
      PRIMARY KEY (idCurso));

-- TABELA EVENTO
CREATE TABLE IF NOT EXISTS Evento (
      idEvento INT NOT NULL AUTO_INCREMENT,
      nomeEvento VARCHAR(20) NOT NULL,
      carga_horaria INT NOT NULL,
      dataInicio DATE NOT NULL,
      dataFim DATE NOT NULL,
    -- Define idEvento como chave primária
      PRIMARY KEY (idEvento));

-- TABELA FUNCIONARIO
CREATE TABLE IF NOT EXISTS Funcionario (
      idFuncionário INT NOT NULL AUTO_INCREMENT,
      nomeFuncionario VARCHAR(20) NOT NULL,
      cargoFuncionario VARCHAR(20) NOT NULL,
      Empresa_idEmpresa INT NOT NULL,
    -- Define idFuncionário como chave primária,
      PRIMARY KEY (idFuncionário),
      INDEX fk_Funcionário_Empresa1_idx (Empresa_idEmpresa ASC) VISIBLE,
    -- Define Empresa_idEmpresa, como chave estrangeira
      CONSTRAINT fk_Funcionário_Empresa1
            FOREIGN KEY (Empresa_idEmpresa)
            REFERENCES Empresa (idEmpresa)
            ON DELETE RESTRICT
            ON UPDATE NO ACTION);

-- TABELA LICENCIA
CREATE TABLE IF NOT EXISTS Licencia (
      idProfessor INT NOT NULL,
      idCurso INT NOT NULL,
      -- Define idProfessor e idCurso como chave primária
      PRIMARY KEY (idProfessor, idCurso),
```

```
      INDEX fk_Professor_has_Curso_Curso1_idx (idCurso ASC) VISIBLE,
    -- Define idProfessor, como chave estrangeira
      CONSTRAINT fk_Professor_has_Curso_Professor
            FOREIGN KEY (idProfessor)
            REFERENCES Professor (idProfessor)
            ON DELETE CASCADE
            ON UPDATE NO ACTION,
      -- Define idCurso, como chave estrangeira
      CONSTRAINT fk_Professor_has_Curso_Curso1
            FOREIGN KEY (idCurso)
            REFERENCES Curso (idCurso)
            ON DELETE CASCADE
            ON UPDATE NO ACTION);

-- TABELA COORDENA
CREATE TABLE IF NOT EXISTS Coordena (
      idEvento INT NOT NULL,
      idProfessor INT NOT NULL,
    -- Define idEvento e idProfessor como chave primária
      PRIMARY KEY (idEvento, idProfessor),
      INDEX fk_Evento_has_Professor_Professor1_idx (idProfessor ASC) VISIBLE,
      INDEX fk_Evento_has_Professor_Evento1_idx (idEvento ASC) VISIBLE,
      -- Define idEvento, como chave estrangeira
      CONSTRAINT fk_Evento_has_Professor_Evento1
            FOREIGN KEY (idEvento)
            REFERENCES Evento (idEvento)
            ON DELETE CASCADE
            ON UPDATE NO ACTION,
      -- Define idProfessor, como chave estrangeira
      CONSTRAINT fk_Evento_has_Professor_Professor1
            FOREIGN KEY (idProfessor)
            REFERENCES Professor (idProfessor)
            ON DELETE RESTRICT
            ON UPDATE NO ACTION);

-- TABELA PARTICIPA
CREATE TABLE IF NOT EXISTS Participa (
      idEvento INT NOT NULL,
      idFuncionário INT NOT NULL,
      -- Define idEvento e idFuncionário como chave primária
      PRIMARY KEY (idEvento, idFuncionário),
      INDEX fk_Evento_has_Funcionário_Funcionário1_idx (idFuncionário ASC) VISIBLE,
      INDEX fk_Evento_has_Funcionário_Evento1_idx (idEvento ASC) VISIBLE,
    -- Define idEvento, como chave estrangeira
      CONSTRAINT fk_Evento_has_Funcionário_Evento1
```

```
            FOREIGN KEY (idEvento)
            REFERENCES Evento (idEvento)
            ON DELETE CASCADE
            ON UPDATE NO ACTION,
      -- Define idFuncionário, como chave estrangeira
      CONSTRAINT fk_Evento_has_Funcionário_Funcionário1
            FOREIGN KEY (idFuncionário)
            REFERENCES Funcionario (idFuncionário)
            ON DELETE CASCADE
            ON UPDATE NO ACTION);

-- TABELA MATRICULA
CREATE TABLE IF NOT EXISTS Matricula (
      progresso INT DEFAULT 0,
      dataInicio DATE NOT NULL,
      idCurso INT NOT NULL,
      idAluno INT NOT NULL,
      certificado VARCHAR(1) DEFAULT 'N',
      -- Define dataInicio, idCurso e idAluno como chave primária
      PRIMARY KEY (dataInicio, idCurso, idAluno),
      INDEX fk_Matricula_Curso1_idx (idCurso ASC) VISIBLE,
      INDEX fk_Matricula_Aluno1_idx (idAluno ASC) VISIBLE,
      -- Define idCurso, como chave estrangeira
      CONSTRAINT fk_Matricula_Curso1
            FOREIGN KEY (idCurso)
            REFERENCES Curso (idCurso)
            ON DELETE CASCADE
            ON UPDATE NO ACTION,
      -- Define idAluno, como chave estrangeira
      CONSTRAINT fk_Matricula_Aluno1
            FOREIGN KEY (idAluno)
            REFERENCES Aluno (idAluno)
            ON DELETE CASCADE
            ON UPDATE NO ACTION);

-- TABELA HORA ACESSO
CREATE TABLE IF NOT EXISTS HoraAcesso (
      idEvento INT NOT NULL,
      idFuncionario INT NOT NULL,
      dataAcesso DATE NOT NULL,
      horarioInicial TIME NOT NULL,
      horarioFinal TIME NULL,
     -- Define dEvento,idFuncionario, dataAcesso, horarioInicial
      PRIMARY KEY (idEvento, idFuncionario, dataAcesso, horarioInicial),
     -- Define idEvento e dFuncionário como chave estrangeira
```

```
     CONSTRAINT fk_horaAcesso_participa1
          FOREIGN KEY (idEvento , idFuncionario)
          REFERENCES Participa (idEvento , idFuncionário)
          ON DELETE CASCADE
          ON UPDATE NO ACTION);


-- TRIGGRER responsável por atualizar o atributo Certificado na tabela Matricula
DELIMITER $$
CREATE TRIGGER trg_Certificado
BEFORE UPDATE ON Matricula
 FOR EACH ROW
 BEGIN
     IF NEW.progresso >=60 THEN
          SET NEW.certificado = 'S';
     END IF;
 END $$
DELIMITER ;


commit;
```

b)

```
USE aluraDb;
/*Adiciona na tabela Aluno uma coluna que indica a quantidade de cursos feitos */
ALTER TABLE Aluno ADD COLUMN qtdCursos INT NULL;
SELECT *
FROM Aluno;

/*Remove na tabela Aluno a coluna qtdCursos*/
ALTER TABLE Aluno DROP COLUMN qtdCursos CASCADE;
SELECT *
FROM Aluno;
##############



/*Cria a tabela Patrocinador*/
CREATE TABLE Patrocinador(-- cria Tabela Patrocinador
idPatrocinador INT NOT NULL,
PRIMARY KEY (idPatrocinador)
);
```

```
/*Cria a coluna idPat (id do patrocinador) na tabela Evento*/
ALTER TABLE Evento ADD COLUMN idPat INT NULL;

/*adiciona o idPatatrocinador como chave estrangeira na tabela Evento*/
ALTER TABLE Evento ADD CONSTRAINT
fk_par FOREIGN KEY (idPat) REFERENCES
Patrocinador (idPatrocinador);
SELECT *
FROM Evento;

/*Remove a relacao de chave estrangeira de Patrocinador na tabela Evento*/
ALTER TABLE Evento
DROP CONSTRAINT fk_par;

/*Remove a coluna idPat da tabela Evento*/
ALTER TABLE Evento DROP COLUMN idPat CASCADE;
SELECT *
FROM Evento;

/*Exclui a tabela Patrocinadores*/
DROP TABLE Patrocinador CASCADE;
##############

/*Altera a Tabela HoraAcesso acrescentando hora total de participacao em eventos*/
SELECT * FROM HoraAcesso;
ALTER TABLE HoraAcesso ADD COLUMN hora_part_evento TIME NULL;

/*Atualiza todas as horas de participacao em eventos*/
UPDATE HoraAcesso
SET hora_part_evento =SEC_TO_TIME(TIME_TO_SEC(horarioFinal) -
TIME_TO_SEC(horarioInicial));
SELECT * FROM HoraAcesso;

/*Cria um TRIGGER que atualiza a cada update as horas de acesso*/
DELIMITER $$
CREATE TRIGGER trg_HoraTotal
BEFORE UPDATE ON HoraAcesso
 FOR EACH ROW
 BEGIN
     IF NEW.horarioFinal !=old.horarioFinal OR NEW.horarioInicial !=
OLD.horarioInicial THEN
          SET NEW.hora_part_evento =SEC_TO_TIME(TIME_TO_SEC(NEW.horarioFinal) -
TIME_TO_SEC(NEW.horarioInicial));
    END IF;
 END $$
```

```
DELIMITER ;

/*Exemplo de update*/
UPDATE HoraAcesso
SET horarioFinal = "15:00:00"
WHERE idFuncionario = 20;
SELECT * FROM HoraAcesso;

/*Exclui o TRIGGER HoraTotal*/
DROP TRIGGER trg_HoraTotal;

/*Remove hora de participao da tabela HoraAcesso*/
ALTER TABLE HoraAcesso DROP COLUMN hora_part_evento;
SELECT * FROM HoraAcesso;
###############
```

c)

```
use aluraDb;

/* c) exemplos de inserção de dados em todas as tabelas */

-- exemplo de inserção na tabela Professor

INSERT INTO Professor VALUES (NULL, "Mario Souto", "Engenharia de Software",
2000.00, "M");
INSERT INTO Professor VALUES (NULL, "Sara Malvar", "Data Science", 120000.00, "F");
INSERT INTO Professor VALUES (NULL, "Paulo Silveira", "Java", 1000.00, "M");
INSERT INTO Professor VALUES (NULL, "Guilherme Silveira", "Engenharia de Software",
2000.00, "M");
INSERT INTO Professor VALUES (NULL, "Jualiana Negreiros", "Front End", 23000.00,
"F");
INSERT INTO Professor VALUES (NULL, "Priscila Stuani", "Comportamento e soft
skills", 21000.00, "F");
INSERT INTO Professor VALUES (NULL, "Roberta Arcoverde", "Engenharia de Software",
20000.00, "F");
INSERT INTO Professor VALUES (NULL, "Loiane Groner", "Engenharia de Software",
15000.00, "F");
INSERT INTO Professor VALUES (NULL, "Gustavo Guanabara", "Engenharia de Software",
50000.00, "M");
INSERT INTO Professor VALUES (NULL, "Marco Bruno", "Front End", 12200.00, "M");
INSERT INTO Professor VALUES (NULL, "Fabio Akita", "Youtuber", 2100.00, "M");
INSERT INTO Professor VALUES (NULL, "Filipe Deschamps", "Engenharia de Software",
5000.00, "M");
INSERT INTO Professor VALUES (NULL, "Mayk Brito", "Full Stack", 12000.00,"M");
```

```sql
INSERT INTO Professor VALUES (NULL, "Diego Fernandes", "Full Stack", 12000.00,"M");
INSERT INTO Professor VALUES (NULL, "Rodrigo Gonçalves", "Engenharia de Software",
1000.00,"M");
INSERT INTO Professor VALUES (NULL, "Alfredo Junior", "Hardware", 2000.00,"M");
INSERT INTO Professor VALUES (NULL, "Anderson Gaveta", "Edicao de video",
2000.00,"M");
INSERT INTO Professor VALUES (NULL, "Maria Gabriela Oliveira", "SQL", 7000.00,"F");
INSERT INTO Professor VALUES (NULL, "Guilherme Lima", "Engenharia de Software",
5000.00,"M");
INSERT INTO Professor VALUES (NULL, "Janaina Oliveira", "Engenharia de Software",
30000.00,"F");
INSERT INTO Professor VALUES (NULL, "Jubileu Nunes Beraldo", "Youtuber", 960.58,
'M');
INSERT INTO Professor VALUES (NULL, "Carlos Sanfoneu Junior", "DBA", 10600, 'M');
INSERT INTO Professor VALUES (NULL, "João Malaquias Pires", "Engenheiro de DevOps",
9960.58, 'M');
INSERT INTO Professor VALUES (NULL, "Julio Lima de Abreu", "Back-End", 15860, 'M');


-- exemplo de inserção na tabela Aluno

INSERT INTO Aluno VALUES (NULL,"Paulo Barreto", "M");
INSERT INTO Aluno VALUES (NULL, "Ronald Rios", "M");
INSERT INTO Aluno VALUES (NULL, "Juninho da Shineray", "M");
INSERT INTO Aluno VALUES (NULL, "Carimbo da Silva", "M");
INSERT INTO Aluno VALUES (NULL, "Copia Autenticada Francisco Silva", "M");
INSERT INTO Aluno VALUES (NULL, "Rafael Kyoko", "M");
INSERT INTO Aluno VALUES (NULL, "Arthur Valentim", "M");
INSERT INTO Aluno VALUES (NULL, "Anderson Guilherme", "M");
INSERT INTO Aluno VALUES (NULL, "Lorenzo Duarte", "M");
INSERT INTO Aluno VALUES (NULL, "Kevin Michael", "M");
INSERT INTO Aluno VALUES (NULL, "Sophia Juliano", "F");
INSERT INTO Aluno VALUES (NULL, "Estela Frederico", "F");
INSERT INTO Aluno VALUES (NULL, "Isadora Sabrina", "F");
INSERT INTO Aluno VALUES (NULL, "Mariana José", "F");
INSERT INTO Aluno VALUES (NULL, "Maitê Amanda", "F");
INSERT INTO Aluno VALUES (NULL, "Beatriz Augusto", "F");
INSERT INTO Aluno VALUES (NULL, "Bianca Kelvin", "F");
INSERT INTO Aluno VALUES (NULL, "Alícia Lucca", "F");
INSERT INTO Aluno VALUES (NULL, "Mateus Henrique", "M");
INSERT INTO Aluno VALUES (NULL, "Anabela Enzo", "F");


-- exemplos de inserção na tabela Empresa
```

```sql
INSERT INTO Empresa VALUES (NULL, "Jovem Nerd");
INSERT INTO Empresa VALUES (NULL, "Não Salvo");
INSERT INTO Empresa VALUES (NULL, "Gaveta filmes");
INSERT INTO Empresa VALUES (NULL, "Extra");
INSERT INTO Empresa VALUES (NULL, "Porta dos Fundos");
INSERT INTO Empresa VALUES (NULL, "Mercado Livre");
INSERT INTO Empresa VALUES (NULL, "Amazon");
INSERT INTO Empresa VALUES (NULL, "Ponto Frio");
INSERT INTO Empresa VALUES (NULL, "Americanas");
INSERT INTO Empresa VALUES (NULL, "Carrefour");
INSERT INTO Empresa VALUES (NULL, "Nubank");
INSERT INTO Empresa VALUES (NULL, "Rede Tv");
INSERT INTO Empresa VALUES (NULL, "Rede Globo");
INSERT INTO Empresa VALUES (NULL, "Rede Bandeirantes");
INSERT INTO Empresa VALUES (NULL, "Liberty Media");
INSERT INTO Empresa VALUES (NULL, "Hotmart");
INSERT INTO Empresa VALUES (NULL, "Magalu");
INSERT INTO Empresa VALUES (NULL, "Rocketseat");
INSERT INTO Empresa VALUES (NULL, "Udemy");
INSERT INTO Empresa VALUES (NULL, "Shopee");
INSERT INTO Empresa VALUES (NULL, "Google");
INSERT INTO Empresa VALUES (NULL, "Oracle");
INSERT INTO Empresa VALUES (NULL, "Microsoft");
INSERT INTO Empresa VALUES (NULL, "Smart Consulting");


-- exemplos de inserção na tabela Curso

INSERT INTO Curso VALUES (NULL, "Arquitetura e Design de Projetos Java", "Plano
mensal");
INSERT INTO Curso VALUES (NULL, "Clojure", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Elixir", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Engenharia de Software", "Plano mensal");
INSERT INTO Curso VALUES (NULL, "Go Language", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Internet das Coisas", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Jogos com Java", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "JavaScript para back-end", "Plano mensal");
INSERT INTO Curso VALUES (NULL, "Linguagem C++", "Plano mensal");
INSERT INTO Curso VALUES (NULL, "Kotlin", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "PHP", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "TypeScript", "Plano mensal");
INSERT INTO Curso VALUES (NULL, "HTML", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Svelte", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "JavaScript para Front-end", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "React com TypeScript", "Plano Anual");
```

```sql
INSERT INTO Curso VALUES (NULL, "Deep Learning com Pytorch", "Plano mensal");
INSERT INTO Curso VALUES (NULL, "SQL com PostgreSQL", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Figma", "Plano Anual");
INSERT INTO Curso VALUES (NULL, "Produção com Photoshop", "Plano Anual");


-- exemplos de inserção na tabela evento

INSERT INTO Evento VALUES (NULL, "Imersao Dev", 20, "2019-11-30", "2020-01-28");
INSERT INTO Evento VALUES (NULL, "Semcomp", 15, "2019-12-30", "2020-01-18");
INSERT INTO Evento VALUES (NULL, "Imersao BD", 10, "2019-09-10", "2020-05-27");
INSERT INTO Evento VALUES (NULL, "CS50", 15, "2022-01-10", "2022-01-28");
INSERT INTO Evento VALUES (NULL, "SE71", 177, "2019-02-10", "2019-02-28");
INSERT INTO Evento VALUES (NULL, "CompWeek", 20, "2019-01-15", "2020-01-05");
INSERT INTO Evento VALUES (NULL, "Imersao JavaScript", 51, "2018-06-17",
"2019-12-05");
INSERT INTO Evento VALUES (NULL, "Imersao UX/Design", 19, "2019-03-10",
"2019-04-10");
INSERT INTO Evento VALUES (NULL, "Imersao Markting", 100, "2020-02-10",
"2020-03-04");
INSERT INTO Evento VALUES (NULL, "Imersao SEO", 122, "2020-01-10", "2020-01-11");
INSERT INTO Evento VALUES (NULL, "Imersao C++", 11,"2022-05-10", "2022-05-28");
INSERT INTO Evento VALUES (NULL, "Imersao M. Design", 20, "2020-01-10",
"2021-01-12");
INSERT INTO Evento VALUES (NULL, "Imersao Storytelling", 19, "2022-06-10",
"2022-07-07");
INSERT INTO Evento VALUES (NULL, "Imersao Front-end", 10, "2021-07-10",
"2021-07-13");
INSERT INTO Evento VALUES (NULL, "Imersao Back-end", 110, "2022-02-10",
"2022-02-14");
INSERT INTO Evento VALUES (NULL, "Imersao IoT", 12, "2021-04-07", "2021-04-09");
INSERT INTO Evento VALUES (NULL, "Imersao Full-Stack", 11, "2019-06-10",
"2019-07-28");
INSERT INTO Evento VALUES (NULL, "Imersao Java", 60, "2020-01-10", "2020-02-03");
INSERT INTO Evento VALUES (NULL, "Imersao SQL", 121, "2019-11-10", "2019-12-31");
INSERT INTO Evento VALUES (NULL, "Imersao Figma", 11, "2018-01-05", "2018-01-09");


-- exemplos de inserção na tabela Funcionario

INSERT INTO Funcionario VALUES (NULL, "Barreto Paulo ", "Dev Senior", "1");
INSERT INTO Funcionario VALUES (NULL, "Rios Ronald ", "CTO", "1");
INSERT INTO Funcionario VALUES (NULL, "Juninho da Shineray", "Trainee", "3");
INSERT INTO Funcionario VALUES (NULL, "Carimbo da Silva", "Dev Pleno", "1");
INSERT INTO Funcionario VALUES (NULL, "Copia Autenticada", "UX Senior", "3");
```

```sql
INSERT INTO Funcionario VALUES (NULL, "Rafael Kyoko", "Front End junior", "1");
INSERT INTO Funcionario VALUES (NULL, "Rafael Kyoko", "Front End junior", "1");
INSERT INTO Funcionario VALUES (NULL, "Arthur Valentim", "Desenvolvedor", "2");
INSERT INTO Funcionario VALUES (NULL, "Anderson Guilherme", "Full Stack", "2");
INSERT INTO Funcionario VALUES (NULL, "Lorenzo Duarte", "Front End Junior", "1");
INSERT INTO Funcionario VALUES (NULL, "Kevin Michael", "Marketing", "1");
INSERT INTO Funcionario VALUES (NULL, "Sophia Juliano", "Dev Senior", "18");
INSERT INTO Funcionario VALUES (NULL, "Estela Frederico", "Dev Senior", "1");
INSERT INTO Funcionario VALUES (NULL, "Isadora Sabrina", "Dev Senior", "4");
INSERT INTO Funcionario VALUES (NULL, "Mariana José", "Trainee", "5");
INSERT INTO Funcionario VALUES (NULL, "Maitê Amanda", "Dev Senior", "1");
INSERT INTO Funcionario VALUES (NULL, "Beatriz Augusto", "Dev Senior", "2");
INSERT INTO Funcionario VALUES (NULL, "Bianca Kelvin", "Dev Junior", "5");
INSERT INTO Funcionario VALUES (NULL, "Alícia Lucca", "CTO", "6");
INSERT INTO Funcionario VALUES (NULL, "Mateus Henrique", "Junior", "7");
INSERT INTO Funcionario VALUES (NULL, "Carla Alves Santos", "UX Designer", "21");
INSERT INTO Funcionario VALUES (NULL, "Antônio Alves ", "Marketing", "22");
INSERT INTO Funcionario VALUES (NULL, "Renan Costa", "DevOps", "21");
INSERT INTO Funcionario VALUES (NULL, "Ryan Cunha Azevedo", "Diretor", "22");
INSERT INTO Funcionario VALUES (NULL, "Giovana Dias", "Consultor", "21");
INSERT INTO Funcionario VALUES (NULL, "Gabrielle Pereira", "Back end Senior", "21");
INSERT INTO Funcionario VALUES (NULL, "Gustavo Ribeiro", "Motion Design", "22");
INSERT INTO Funcionario VALUES (NULL, "Dorival Marcelo", "Dev Senior", "4");
INSERT INTO Funcionario VALUES (NULL, "Luciano Nine", "Dev Junior", "18");
INSERT INTO Funcionario VALUES (NULL, "Dorina Nina Portinol", "Treinee", "3");
INSERT INTO Funcionario VALUES (NULL, "Camile Ninemati", "Dev Junior", "4");

-- exemplos de inserção na tabela Licencia

INSERT INTO Licencia VALUES ("1", "1");
INSERT INTO Licencia VALUES ("1", "2");
INSERT INTO Licencia VALUES ("1", "8");
INSERT INTO Licencia VALUES ("2", "2");
INSERT INTO Licencia VALUES ("2", "19");
INSERT INTO Licencia VALUES ("3", "12");
INSERT INTO Licencia VALUES ("3", "05");
INSERT INTO Licencia VALUES ("3", "3");
INSERT INTO Licencia VALUES ("4", "1");
INSERT INTO Licencia VALUES ("4", "8");
INSERT INTO Licencia VALUES ("4", "9");
INSERT INTO Licencia VALUES ("5", "10");
INSERT INTO Licencia VALUES ("5", "12");
INSERT INTO Licencia VALUES ("6", "7");
INSERT INTO Licencia VALUES ("6", "8");
INSERT INTO Licencia VALUES ("6", "6");
```

```sql
INSERT INTO Licencia VALUES ("6", "17");
INSERT INTO Licencia VALUES ("7", "18");
INSERT INTO Licencia VALUES ("8", "5");
INSERT INTO Licencia VALUES ("8", "8");
INSERT INTO Licencia VALUES ("9", "4");
INSERT INTO Licencia VALUES ("20", "20");
INSERT INTO Licencia VALUES ("20", "19");
INSERT INTO Licencia VALUES ("10", "10");
INSERT INTO Licencia VALUES ("10", "16");
INSERT INTO Licencia VALUES ("12", "12");
INSERT INTO Licencia VALUES ("12", "11");
INSERT INTO Licencia VALUES ("12", "13");
INSERT INTO Licencia VALUES ("11", "11");
INSERT INTO Licencia VALUES ("13", "13");
INSERT INTO Licencia VALUES ("14", "14");
INSERT INTO Licencia VALUES ("15", "15");
INSERT INTO Licencia VALUES ("16", "16");
INSERT INTO Licencia VALUES ("17", "17");
INSERT INTO Licencia VALUES ("18", "18");
INSERT INTO Licencia VALUES ("19", "19");
INSERT INTO Licencia VALUES ("7", "8");
INSERT INTO Licencia VALUES ("7", "1");
INSERT INTO Licencia VALUES ("7", "9");

-- exemplos de inserção na tabela coordena

INSERT INTO Coordena VALUES ("1", "1");
INSERT INTO Coordena VALUES ("2", "2");
INSERT INTO Coordena VALUES ("3", "3");
INSERT INTO Coordena VALUES ("4", "4");
INSERT INTO Coordena VALUES ("5", "6");
INSERT INTO Coordena VALUES ("6", "5");
INSERT INTO Coordena VALUES ("7", "7");
INSERT INTO Coordena VALUES ("8", "8");
INSERT INTO Coordena VALUES ("9", "9");
INSERT INTO Coordena VALUES ("20", "20");
INSERT INTO Coordena VALUES ("10", "10");
INSERT INTO Coordena VALUES ("12", "12");
INSERT INTO Coordena VALUES ("11", "11");
INSERT INTO Coordena VALUES ("13", "13");
INSERT INTO Coordena VALUES ("14", "14");
INSERT INTO Coordena VALUES ("15", "15");
INSERT INTO Coordena VALUES ("16", "16");
INSERT INTO Coordena VALUES ("17", "17");
INSERT INTO Coordena VALUES ("18", "18");
```

```sql
INSERT INTO Coordena VALUES ("19", "19");


-- exemplos de inserção na tabela Participa

INSERT INTO Participa VALUES ("1", "1");
INSERT INTO Participa VALUES ("1", "2");
INSERT INTO Participa VALUES ("1", "3");
INSERT INTO Participa VALUES ("2", "2");
INSERT INTO Participa VALUES ("2", "4");
INSERT INTO Participa VALUES ("2", "5");
INSERT INTO Participa VALUES ("3", "3");
INSERT INTO Participa VALUES ("4", "6");
INSERT INTO Participa VALUES ("4", "4");
INSERT INTO Participa VALUES ("4", "9");
INSERT INTO Participa VALUES ("4", "7");
INSERT INTO Participa VALUES ("5", "8");
INSERT INTO Participa VALUES ("5", "10");
INSERT INTO Participa VALUES ("5", "12");
INSERT INTO Participa VALUES ("6", "5");
INSERT INTO Participa VALUES ("7", "9");
INSERT INTO Participa VALUES ("7", "15");
INSERT INTO Participa VALUES ("7", "7");
INSERT INTO Participa VALUES ("7", "17");
INSERT INTO Participa VALUES ("8", "8");
INSERT INTO Participa VALUES ("9", "1");
INSERT INTO Participa VALUES ("9", "2");
INSERT INTO Participa VALUES ("9", "3");
INSERT INTO Participa VALUES ("9", "4");
INSERT INTO Participa VALUES ("9", "5");
INSERT INTO Participa VALUES ("9", "9");
INSERT INTO Participa VALUES ("20", "10");
INSERT INTO Participa VALUES ("20", "11");
INSERT INTO Participa VALUES ("20", "12");
INSERT INTO Participa VALUES ("20", "13");
INSERT INTO Participa VALUES ("10", "14");
INSERT INTO Participa VALUES ("12", "1");
INSERT INTO Participa VALUES ("12", "2");
INSERT INTO Participa VALUES ("12", "3");
INSERT INTO Participa VALUES ("12", "6");
INSERT INTO Participa VALUES ("12", "7");
INSERT INTO Participa VALUES ("12", "8");
INSERT INTO Participa VALUES ("12", "9");
INSERT INTO Participa VALUES ("12", "12");
INSERT INTO Participa VALUES ("11", "11");
```

```sql
INSERT INTO Participa VALUES ("13", "3");
INSERT INTO Participa VALUES ("13", "5");
INSERT INTO Participa VALUES ("13", "13");
INSERT INTO Participa VALUES ("14", "11");
INSERT INTO Participa VALUES ("14", "4");
INSERT INTO Participa VALUES ("14", "9");
INSERT INTO Participa VALUES ("14", "3");
INSERT INTO Participa VALUES ("14", "13");
INSERT INTO Participa VALUES ("15", "15");
INSERT INTO Participa VALUES ("16", "16");
INSERT INTO Participa VALUES ("17", "17");
INSERT INTO Participa VALUES ("18", "18");
INSERT INTO Participa VALUES ("18", "17");
INSERT INTO Participa VALUES ("19", "19");
INSERT INTO Participa VALUES ("5", "6");
INSERT INTO Participa VALUES ("20", "20");
INSERT INTO Participa VALUES ("10", "10");
INSERT INTO Participa VALUES ("14", "14");
INSERT INTO Participa VALUES ("2", "12");
INSERT INTO Participa VALUES ("3", "13");
INSERT INTO Participa VALUES ("3", "14");
INSERT INTO Participa VALUES ("3", "8");

-- exemplos de inserção na tabela Matricula

INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 1, 1);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 1, 17);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 1, 15);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 1, 8);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 1, 2);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 3);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 4);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 5);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 6);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 7);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 9);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 11);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 10);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 12);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 5, 13);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 3, 5);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 16);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 2, 15);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 1, 7);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 16, 8);
```

```sql
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 19, 8);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 10, 8);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 13, 2);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 17, 1);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 17, 5);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 17, 8);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 17, 9);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 17, 11);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 13, 14);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 11, 19);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES (CURDATE(), 12, 5);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 10);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 11);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 9);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 5);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 2);
INSERT INTO Matricula (dataInicio, idCurso, idAluno) VALUES(CURDATE(), 8, 3);

-- insercao na tabela hora acesso

INSERT INTO HoraAcesso VALUES ("2", "2", "2022-06-10", "14:23:00", "15:52:10");
INSERT INTO HoraAcesso VALUES ("12", "2", "2022-07-10", "11:23:00", "15:52:10");
INSERT INTO HoraAcesso VALUES ("2", "12", "2022-09-08", "14:23:00", "18:52:10");
INSERT INTO HoraAcesso VALUES ("3", "13", CURDATE(), "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("3", "14", "2022-11-07", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("14", "3", CURDATE(), "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("3", "8", "2022-05-19", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("4", "4", "2022-09-01", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("5", "6", "2022-05-10", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("6", "5", "2019-04-15", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("7", "7", "2019-01-17", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("8", "8", "2019-03-19", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("9", "9", "2019-02-20", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("20", "20", "2019-03-27", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("10", "10", CURDATE(), "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("12", "12", "2018-05-11", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("11", "11", CURDATE(), "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("13", "13", "2019-08-08", "13:23:00", "13:52:10");
INSERT INTO HoraAcesso VALUES ("14", "14", CURDATE(), "13:23:00", "13:52:10");
commit;
```

d)

```sql
USE aluraDb;
SET SQL_SAFE_UPDATES = 0; -- Desabilita o safe update


/*Exibe a tabela antes da atualização*/
SELECT idEvento, nomeProf, dataFim
FROM Professor NATURAL JOIN Coordena NATURAL JOIN Evento
WHERE nomeProf = "Fabio Akita";

-- Atualiza a data do fim do(s) evento(s) cujo o professor Fabio Akita Coordena
UPDATE Evento
SET dataFim = "2020-02-15"
WHERE idEvento IN (
      SELECT idEvento
      FROM Professor NATURAL JOIN Coordena
      WHERE nomeProf = "Fabio Akita"
);

/*Exibe a tabela depois da atualização*/
SELECT idEvento, nomeProf, dataFim
FROM Professor NATURAL JOIN Coordena NATURAL JOIN Evento
WHERE nomeProf = "Fabio Akita";
-- Aumenta mais 4 horas na carga horária de eventos de imersão

/*Exibe a tabela antes da atualizacao*/
SELECT * FROM Evento;

UPDATE Evento
SET carga_horaria = carga_horaria + 4

/*Exibe a tabela depois da atualizacao*/
WHERE nomeEvento LIKE 'Imersao%';
SELECT * FROM Evento;

-- Atualiza o evento em que o professor com id = 3 coordena

/*Exibe a tabela antes da atualização*/
SELECT idEvento, idProfessor, nomeEvento
FROM Coordena NATURAL JOIN Evento
WHERE idProfessor = 3;

UPDATE Coordena
SET idEvento = 2
WHERE idProfessor = 3;
```

```sql
/*Exibe a tabela depois da atualização*/
SELECT idEvento, idProfessor, nomeEvento
FROM Coordena NATURAL JOIN Evento
WHERE idProfessor = 3;

-- Atualiza o plano do curso 'SQL Com PostgreSQL' de anual para semestral

/*Exibe a tabela antes da atualização*/
SELECT plano, nomeCurso
FROM Curso
WHERE nomeCurso = "SQL com PostgreSQL";

UPDATE Curso
SET plano = "Plano Semestral"
WHERE nomeCurso = "SQL com PostgreSQL";

/*Exibe a tabela depois da atualização*/
SELECT plano, nomeCurso
FROM Curso
WHERE nomeCurso = "SQL com PostgreSQL";


-- Atualiza o curso que o professor com id = 13 licencia

/*Exibe a tabela antes da atualizacao*/
SELECT idCurso, idProfessor, nomeCurso
FROM Licencia NATURAL JOIN Curso
WHERE idProfessor = 13;

UPDATE  Licencia
SET idCurso = 8
WHERE idprofessor = 13;

/*Exibe a tabela depois da atualizacao*/
SELECT idCurso, idProfessor, nomeCurso
FROM Licencia NATURAL JOIN Curso
WHERE idProfessor = 13;

-- Aumenta o salário do professor Rodrigo Gonçalves em mais 2560.50

/*Exibe a tabela antes da atualizacao*/
SELECT nomeProf, salario
FROM Professor
WHERE nomeProf = "Rodrigo Gonçalves";
```

```sql
UPDATE Professor
SET salario = salario + 2560.50
WHERE nomeProf  = "Rodrigo Gonçalves";

/*Exibe a tabela depois da atualizacao*/
SELECT nomeProf, salario
FROM Professor
WHERE nomeProf = "Rodrigo Gonçalves";

SET SQL_SAFE_UPDATES = 1; -- Habilita o safe update

/* Safe update é um recurso de atualização segura do Mysql em
   que ele não executa o update ou Delete sem o WHERE com
   uma chave ou LIMIT, esse recurso têm por objetivo evitar
   erros humanos, já que quando não usamos o WHERE, nós
   aplicamos um update em todas al linhas da tabela
*/
```

e)

```sql
use aluraDb;
SET SQL_SAFE_UPDATES= 0;


-- EXEMPLO #1: deletar todos os alunos do sexo feminino que o nome começa com a
Letra A

-- SELECT mostrando alunos do sexo feminino com a letra A
SELECT nomeAluno FROM Aluno WHERE sexo = "F" AND nomeAluno LIKE "A%";

-- DELETE
DELETE FROM Aluno
WHERE sexo = "F" AND nomeAluno LIKE "A%";

-- SELECT mostrando que não há mais alunos do sexo feminino com a letra A
SELECT nomeAluno FROM Aluno WHERE sexo = "F" AND nomeAluno LIKE "A%";

-- FIM EXEMPLO #1;
```

```sql
-- EXEMPLO #2: deletar todos os funcionarios que trabalham na empresa Amazon

-- SELECT retorna todos os funcionarios que trabalham na empresa Amazon
SELECT nomeFuncionario
FROM Funcionario
WHERE Empresa_idEmpresa IN (SELECT idEmpresa
                                   FROM Empresa
                            WHERE nomeEmpresa = "Amazon");

-- deleta todos os funcionarios que trabalham na Amazon
DELETE FROM Funcionario
WHERE Empresa_idEmpresa IN (SELECT idEmpresa
                                   FROM Empresa
                            WHERE nomeEmpresa = "Amazon");


-- SELECT monstrando que foram todos apagados
SELECT nomeFuncionario FROM Funcionario
WHERE Empresa_idEmpresa IN (SELECT idEmpresa
                                   FROM Empresa
                            WHERE nomeEmpresa = "Amazon");


-- FIM EXEMPLO #2;


-- EXEMPLO #3: Deleta todas as empresas que não possuem nenhum funcionário

-- Tabela antes do delete, indicando todas as empresas, inclusives as que nao
possuem funcionarios
SELECT *
FROM Empresa LEFT OUTER JOIN Funcionario ON idEmpresa = Empresa_idEmpresa;

-- Deleta todas as empresas que nao possuem nenhum funcionario
DELETE FROM Empresa
WHERE idEmpresa NOT IN(
      SELECT idEmpresa FROM(
      SELECT idEmpresa
      FROM Empresa INNER JOIN Funcionario ON idEmpresa = Empresa_idEmpresa) AS idE
);

-- Tabela após delete
SELECT *
FROM Empresa LEFT OUTER JOIN Funcionario ON idEmpresa = Empresa_idEmpresa;
```

```
-- FIM EXEMPLO #3;


-- EXEMPLO #4: deleta os Eventos que possuem a palavra imersao no nome

SELECT nomeEvento FROM Evento WHERE nomeEvento LIKE "imersao%";

-- DELETE
DELETE FROM Evento
WHERE nomeEvento LIKE "imersao%";

-- SELECT mostrando que não há mais eventos desse tipo
SELECT nomeEvento FROM Evento WHERE nomeEvento LIKE "imersao%";

-- FIM EXEMPLO #4


-- EXEMPLO #5: deleta todas as matrículas do curso "JavaScript para back-end"

-- SELECT para retornar o id de alunos matriculados no curso
SELECT idAluno FROM Matricula
WHERE idCurso IN (SELECT idCurso
                    FROM Curso
                  WHERE nomeCurso = "JavaScript para back-end");

-- DELETE de todas as matriculas
DELETE FROM Matricula
WHERE idCurso IN (SELECT idCurso
                    FROM Curso
                  WHERE nomeCurso = "JavaScript para back-end");

SELECT idAluno FROM Matricula
WHERE idCurso IN (SELECT idCurso
                    FROM Curso
                  WHERE nomeCurso = "JavaScript para back-end");

-- FIM EXEMPLO #5
```

f)

```
USE aluraDb;

/*SELECT, FROM, WHERE,
```

```
ORDER BY, GROUP BY, HAVING*/

/* 1- Seleciona em ordem crescente todas as empresas e soma da carga horaria total
de todos os eventos que seus funcionarios estao cadastrados */
SELECT idEmpresa, nomeEmpresa, SUM(carga_horaria) AS CargaHorariaTotal
FROM Evento NATURAL JOIN Participa NATURAL JOIN Funcionario
INNER JOIN Empresa ON Empresa_idEmpresa = idEmpresa
GROUP BY Empresa_idEmpresa
ORDER BY CargaHorariaTotal ASC;


################


/* 2- Seleciona as empresas que possuem mais de 2 funcionarios*/
SELECT Empresa_idEmpresa,nomeEmpresa, COUNT(*) AS qtdeFuncionarios
FROM Funcionario INNER JOIN Empresa ON Empresa_idEmpresa=idEmpresa
GROUP BY Empresa_idEmpresa
HAVING qtdeFuncionarios >2;
###############


/*3 -Seleciona o nome todos os alunos aprovados no ano de 2019*/
SELECT nomeAluno, progresso
FROM Aluno NATURAL JOIN Matricula
WHERE progresso >=60 AND DataInicio BETWEEN "2019-01-01" AND " 2019-12-31";
###############

/*JOIN, OUTER JOIN,UNION*/
-- 4- Recupere o nome e salario dos professores do sexo masculino ou que ministra o
curso de banco de dados

SELECT nomeProf, salario
FROM Professor
WHERE sexo = "M" OR
      idProfessor IN (
                  SELECT idProfessor
            FROM Licencia NATURAL JOIN Curso
            WHERE nomeCurso = "Banco de Dados"
            );


-- 5- Selecione o nome dos funcionários que trabalham na empresa de nome 'Google' ou
que trabalham na empresa de nome 'Oracle'.
```

```sql
select nomeFuncionario AS Nome_Funcionario
from Funcionario F join Empresa E
ON F.Empresa_idEmpresa = E.idEmpresa
where nomeEmpresa='Google'
union
select nomeFuncionario AS Nome_Funcionario
from Funcionario F join Empresa E
ON F.Empresa_idEmpresa = E.idEmpresa
where nomeEmpresa='Oracle';


-- 6- Recupera os cursos que a professora Roberta Arcoverde licencia

SELECT nomeCurso AS nome_do_curso
FROM Curso C JOIN Licencia L
ON C.idCurso = L.idCurso
JOIN Professor P ON P.idProfessor = L.idProfessor
WHERE nomeProf = 'Roberta Arcoverde';

-- 7- Recuperar os dados de cada aluno e de suas respectivas matrículas, se o aluno
ainda não estiver feito nenhuma matrícula indique com valores nulos

SELECT * FROM
Aluno A LEFT OUTER JOIN Matricula M
ON A.idALuno = M.idAluno;

-- left outer join mantém todas as tuplas da tabela aluno na consulta
-- As tuplas da tabela Matricula podem ser nulas se aluno ainda não estiver se
matriculado em nenhum curso

-- nome e salario dos professores que licenciam o curso de id = 1111, ou que
ministram evento de id = 789

/*AND, OR, NOT, BETWEEN*/

/*Seleciona os professores homens que recebem entre 1 a 5mil em ordem crescente*/
SELECT idProfessor, nomeProf, salario
FROM Professor
WHERE sexo = 'M' AND salario BETWEEN 1000 AND 5000
ORDER BY salario ASC;

/*Recupera os cursos que nao há nenhuma matricula*/
SELECT nomeCurso
FROM Curso
WHERE idCurso NOT IN(SELECT idCurso FROM Matricula);
```

```sql
/*IN, LIKE, IS NULL, ANY/SOME, ALL, EXISTS */
-- 8- Retorna o nome e o salário(ordem decrescente) do professor(es) que possuiu o
-- salário maior que média do salário de todos os professores da plataforma

SELECT nomeProf AS "Nome do Professor", salario AS "Salário"
FROM Professor
WHERE salario > SOME (
      SELECT AVG(salario)
      FROM Professor)
ORDER BY salario DESC;

-- 9- Recupera nome dos Alunos que não possuem Matrícula

SELECT nomeAluno AS "Aluno que não\n possuem matrícula"
FROM Aluno
WHERE idAluno NOT IN (
      SELECT idAluno
      FROM Matricula
);

-- 10- Recupera o total de eventos de imersão e média da carga entre eles.

SELECT COUNT(nomeEvento) AS "Qtde total de\nEventos de Imersão",
         AVG(carga_horaria) AS "Média de Carga\nHorária"
FROM Evento
WHERE nomeEvento LIKE "Imersão%";

-- 11- Recupera o em ordem alfabética nome do(s) que não coordenam nenhum Evento.

SELECT nomeProf AS "Professores que não\ncoordenam nenhum evento"
FROM Professor P LEFT OUTER JOIN Coordena C
ON P.idProfessor = C.idProfessor
WHERE C.idEvento IS NULL
ORDER BY nomeProf;

-- 12- Recupera o nome dos Funcionários que trabalham na
-- empresa com id = 1 e estão na posição de Senior

SELECT F.nomeFuncionario
FROM Funcionario F
WHERE EXISTS(
      SELECT *
```

```
    FROM Empresa G
    WHERE G.idEmpresa = F.Empresa_idEmpresa
           AND G.idEmpresa = 1
        AND F.cargoFuncionario LIKE "%Senior%"
);

-- 13- Recupera o nome e o ID dos Funcinários que trabalham na mesma empresa que o
"Arthur Valentim"
-- Incluindo o próprio "Arthur Valentim"

SELECT nomeFuncionario AS "Funcionário", idFuncionário AS "ID"
FROM Funcionario
WHERE Empresa_idEmpresa = ALL (
     SELECT Empresa_idEmpresa
    FROM Funcionario F
    WHERE nomeFuncionario = "Arthur Valentim"
);
```

g)

```
USE aluraDb;

/*Cria uma visao entre a tabela Funcionario, Empresa e Participam, que indica todos
os participantes de eventos */
CREATE VIEW Participantes AS
SELECT nomeEvento, idFuncionário, nomeFuncionario, cargoFuncionario
FROM Evento NATURAL JOIN
 Funcionario NATURAL JOIN Participa;
SELECT *
FROM Participantes;
/*Utilzia a tabela Participantes para contar quantos funcionários participaram de um
evento*/
SELECT nomeEvento, COUNT(*) AS qtdeParticipantes
FROM Participantes
GROUP BY nomeEvento;
#############
DROP VIEW Participantes;


/*Cria uma visao de todos os nomes do cursos, e seus alunos*/
CREATE VIEW matriculaCursos AS
SELECT idCurso, nomeCurso, idAluno, nomeAluno
FROM Curso NATURAL JOIN Matricula;
```

```sql
/*Ordena os cursos pela maior quantidade de alunos*/
SELECT idCurso, nomeCurso, COUNT(*) AS qtdAlunos
FROM matriculaCursos
GROUP BY idCurso
ORDER BY qtdAlunos ASC;
#############
DROP VIEW matriculaCursos;


/*Cria uma visão de todos os alunos de um professor */
CREATE VIEW Alunos_Matriculados AS
SELECT L.idProfessor, idAluno, progresso
FROM Aluno NATURAL JOIN
Matricula NATURAL JOIN
Curso NATURAL JOIN
Licencia L INNER JOIN Professor P ON L.idProfessor = P.idProfessor ;

/*Ultiliza a tabela Alunos_Matriculados para contar a quantidade de alunos que um
professor poussui e a media de seus alunos*/
SELECT idProfessor, COUNT(*) AS qtdAlunos, AVG(progresso) AS Media
FROM Alunos_Matriculados
GROUP BY idProfessor;
#############
DROP VIEW Alunos_Matriculados;
```

h)

```sql
-- Exemplos de criação de usuários concessão e revogação de permissão de acesso
-- Criação e exclusão de um usuário que tem permissão de selecionar dados de todas
as tabela do database aluraDB.
CREATE USER 'devDBA'@'localhost' IDENTIFIED BY '12345678';

GRANT SELECT ON aluraDb.* TO 'devDBA'@'localhost';

DROP USER 'devDBA'@'localhost';

-- Criação e exclusão de um usuário que tem permissão de selecionar
-- dados dos Alunos utilizados para entrega e executar um store producere.

CREATE USER 'davidjc'@'localhost' IDENTIFIED BY '123456';

GRANT SELECT ON aluraDb.Aluno TO 'davidjc'@'localhost';

REVOKE SELECT ON aluraDb.Aluno FROM 'davidjc'@'localhost';
```

```
GRANT EXECUTE ON PROCEDURE aluraDb.CadastraCurso TO 'davidjc'@'localhost';

DROP USER 'davidjc'@'localhost';
```

i)

```
-- PROCEDIMENTO 1

-- Procedimento que realiza o cadastro de um novo curso à grade de cursos da Alura

DELIMITER //
CREATE PROCEDURE CadastraCurso(
   cIdCurso INT,
   cNomeCurso VARCHAR(70),
   cPlano VARCHAR(45)
)
BEGIN
    INSERT INTO Curso
    VALUES (cIdCurso, cNomeCurso, cPlano);
END//
DELIMITER ;

-- Chamada do procedimento CadastraCurso
CALL CadastraCurso(222, "Engenharia de Software", "plus");

-- Select na tabela Curso para verificar se curso foi cadastrado com seucesso
select * from Curso;

-- PROCEDIMETO 2

-- Procedimento que retorna o progresso no curso de determinado aluno pelo seu id

DELIMITER //
CREATE PROCEDURE gerarCertificado(IN aIdAluno INT(9), OUT mProgresso INT)
BEGIN
    select Progresso into mProgresso
    from Matricula M join Curso C
    on M.idCurso = C.idCurso
    where M.Certificado = "N"
    and M.idAluno = aIdAluno;
END //
DELIMITER ;

CALL gerarCertificado(11, @progresso);
```

```sql
select @progresso;

--  PROCEDIMENTO 3

-- Procedimento concede selo para alunos que estão matriculados em mais de 10 cursos
-- É necessário adicionar a coluna selo à tabela Matrícula

ALTER TABLE Matricula
ADD selo VARCHAR(1) NULL DEFAULT"N";

-- Coluna selo adicionada com valor default "N"

DELIMITER //
CREATE PROCEDURE concederSelo(id INT(9))
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE contador INT DEFAULT 0;
    DECLARE idDoAluno INT(9);
    DECLARE meuCursor CURSOR FOR SELECT idAluno FROM Matricula;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET i = 1;
    OPEN meuCursor;
    WHILE(i != 1) DO
        FETCH meuCursor INTO idDoAluno;
        IF idDoAluno = id THEN
            SET contador = contador + 1;
        END IF;
        IF contador >= 10 THEN
            UPDATE Matricula SET selo = "S";
        END IF;
    END WHILE;
    CLOSE meuCursor;
END //
DELIMITER ;

CALL concederSelo(1);
```