Integrantes do grupo:

Bryan de Lima Naneti Barbosa

Enzo Velo

Heitor Rodrigues Sabino

Lucas Gomes Colombo

Rafael Brunini Pereira

Ranulfo Mascari Neto

Descrição:

Levantamento de dados para um sistema acadêmico de uma faculdade.

A faculdade é organizada funcionalmente em departamentos, os quais possuem um nome (chave), uma unidade acadêmica, um telefone e um endereço (constituído de logradouro, bairro, cidade, estado). Existem diversos cursos que pertencem a uma faculdade, sendo que cada curso pertence a um único departamento, enquanto um departamento pode ter vários cursos. Dos cursos, é importante armazenar os atributos código (chave), nome, quantidade de períodos, período (o qual a matéria pertence), modalidade (presencial ou ead).

O curso de um departamento ainda possui diversas disciplinas lecionadas na faculdade. Essas disciplinas possuem um nome, um código (chave), os créditos e o número de horas práticas e teóricas. Um curso pode ter diversas disciplinas diferentes da mesma forma que uma disciplina pode ser de mais de um curso diferente.

Em uma faculdade existem um conjunto de pessoas. Cada pessoa possui uma data de nascimento, um sexo, um CPF (chave), um nome social, um telefone (multivalorado que também pode não receber nenhum valor), um endereço constituído de logradouro, número, complemento, bairro, cidade e estado e uma idade (calculada com os dados da sua data de nascimento até o momento presente).

As pessoas podem ser classificadas como aluno ou professor (ou ambos ao mesmo tempo). Um aluno tem uma matrícula (chave), um e-mail (chave), um período (calculado a partir do período inicial e a data da turma mais recente matriculada), uma data de admissão, um período inicial, e uma situação (se o aluno está matriculado ou não, e é calculado pela sua data de admissão e matrícula). Já um professor tem seu registro (chave), e-mail (chave) e área de interesse (podem haver mais de uma).

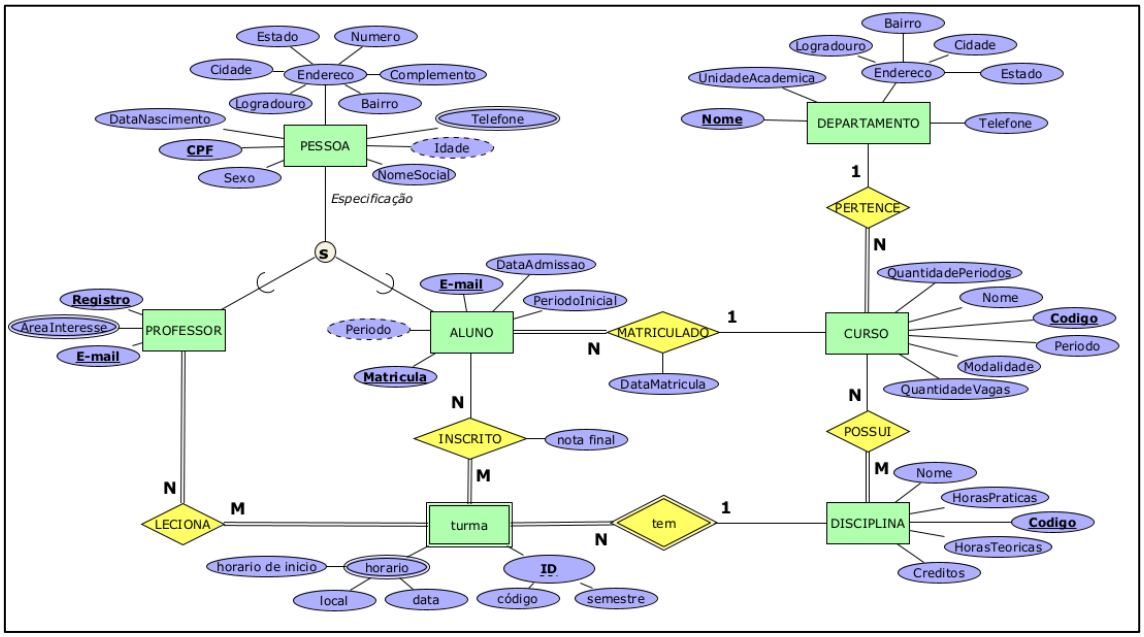
Um aluno está inscrito em uma turma que possui um horário (multivalorado e composto por local e data) de cada turma, e um ID (chave secundária e constituído do código da turma e o semestre dela). Um aluno pode se inscrever em mais de uma turma, da mesma forma que uma turma pode ter mais de um aluno inscrito nela. É importante saber a nota final que cada aluno teve nesta turma.

Um professor também está relacionado a uma turma, sendo que um professor pode lecionar mais de uma turma e uma turma poderá ser lecionada para mais de um professor.

Um aluno deve estar relacionado a um curso de forma que um aluno possa estar matriculado em apenas um curso por vez ao passo que um curso pode ter diversos alunos. Nessa relação é importante armazenar a data de matrícula dos alunos no curso em questão.

Uma turma também deve ter uma disciplina, sendo que uma turma deve ter apenas uma disciplina e uma disciplina pode ter diversas turmas.

Diagrama ER:



Dicionário de Dados:

Tipo Entidade	Pessoa		
Descrição	Conjunto das pessoas que fazem parte da universidade		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nome Social	Nome da pessoa	Texto (80)	N
Idade	Idade da pessoa calculada com os dados da sua data de nascimento data atual	Inteiro (2)	N
Sexo	Sexo (masculino, feminino)	Texto (1) M – Masculino F – Feminino	N
CPF	Cadastro de pessoa física	Texto (11) Formato: ddd.ddd.ddd-dd	N
Data de Nascimento	Data de nascimento da pessoa	Data	N
Registro Geral	Registro Geral	Texto (9)	N
Nome do logradouro	Nomenclatura da rua onde mora	Texto (30)	N
Número	Número da casa/apartamento	Texto (4)	N
Nome do bairro	Nomenclatura dada bairro onde mora	Texto (30)	N

Nome da cidade	Nomenclatura da cidade de nascimento	Texto (30)	N
Sigla do estado	Sigla dada ao estado de nascença	Texto (2)	N
Telefone	Número de telefone para contato	Texto (11)	S

Tipo Entidade	Aluno		
Descrição	Conjuntos de alunos matriculados no sistema acadêmico		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Matrícula	Número de matrícula	Texto (9)	N
Período Inicial	Data do semestre letivo de ingresso	Texto (5) Formato: dddd/d	N
Período	Período atual do aluno, calculado a partir do período inicial e a data da turma mais recente matriculada	Inteiro (2)	S
E-mail Institucional	Endereço eletrônico do aluno	Texto (256)	N
Data Admissão	Data de início da graduação	Data	S

Tipo Entidade	Professor		
Descrição	Conjuntos de professores matriculados no sistema acadêmico		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Registro	Registro do professor	Texto (9)	N
E-mail	Endereço eletrônico do professor	Texto (256)	N
Área Interesse	Área em que o professor atua	Texto (80)	S

Tipo Entidade	Curso		
Descrição	Cursos que são ministrados na faculdade		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nome	Nome do curso	Texto (80)	N
Código	Código de identificação do curso	Texto (5) Formato: AAANN	N
Modalidade	Especificação do curso (Bacharelado (b)/Licenciatura(L))	Texto (1)	N
Período	Período no dia em que o curso é ministrado (Integral(I)/Matutino	Texto (1)	N

	(M)/Diurno(D)/Noturno(N))		
Quantidade de Períodos	Quantidade máxima de períodos possíveis	Inteiro (2)	N
Quantidade de Vagas Semestrais	Vagas ofertadas para o curso semestralmente	Inteiro (2)	N

Tipo Entidade	Disciplina		
Descrição	Disciplinas que são ofertadas pelos cursos de graduação		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nome	Nome da disciplina	Texto (80)	N
Código	Código de identificação da disciplina	Texto (5)	N
Créditos	Quantidade de aulas semanais	Inteiro (2)	N
Horas Teóricas	Quantidade de horas teóricas totais	Inteiro (3)	N
Horas Práticas	Quantidade de horas praticas totais	Inteiro (3)	N

Tipo Entidade	Turma		
Descrição	Conjunto de turmas das disciplinas		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Código	Código da turma em questão (Ex: 10A,14A, ...)	Texto (3) Formato: AAN	N
Semestre	Semestre em que a turma é ofertada (Ex: 2022/2)	Texto (5) Formato: dddd/d	N
Horário	Horário do dia em que uma aula é realizada	Texto (60)	N
Horário de inicio	Horário do dia em que a aula se inicia	Texto (5)	N
Data	Dia da semana em que a aula será realizada	Texto (3)	N
Local	Sala e pavilhão em que uma matéria será realizada (Ex: DCC01, PV2-301, etc.)	Texto (6) Formato: AAA-DDD	N

Tipo Entidade	Departamento		
Descrição	Setor aplicado em determinada área de ensino e pesquisa		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**

Unidade Acadêmica	Unidade Acadêmica ao qual o departamento pertence	Texto (80)	N
Nome	Nome do departamento	Texto (80)	N
Telefone	Número de telefone	Inteiro (11)	S
Nome do logradouro	Nomenclatura da rua onde mora	Texto (30)	N
Bairro	Nomenclatura dada bairro onde mora	Texto (30)	N
Cidade	Nomenclatura da cidade de nascimento	Texto (30)	N
Estado	Sigla dada ao estado de nascença	Texto (2)	N

Relacionamentos:

Tipo Relacionamento	Matriculado		
Descrição	Indica em qual curso o aluno está matriculado		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Data de Matrícula	Data em que o aluno foi matriculado no curso	Data	S

Tipo Relacionamento	Inscrito		
Descrição	Indica em qual turma o aluno está		
Atributos			
Nome	**Descrição**	**Domínio**	**Permite nulo? (S/N)**
Nota Final	Nota final obtida na matéria	Inteiro (2)	S

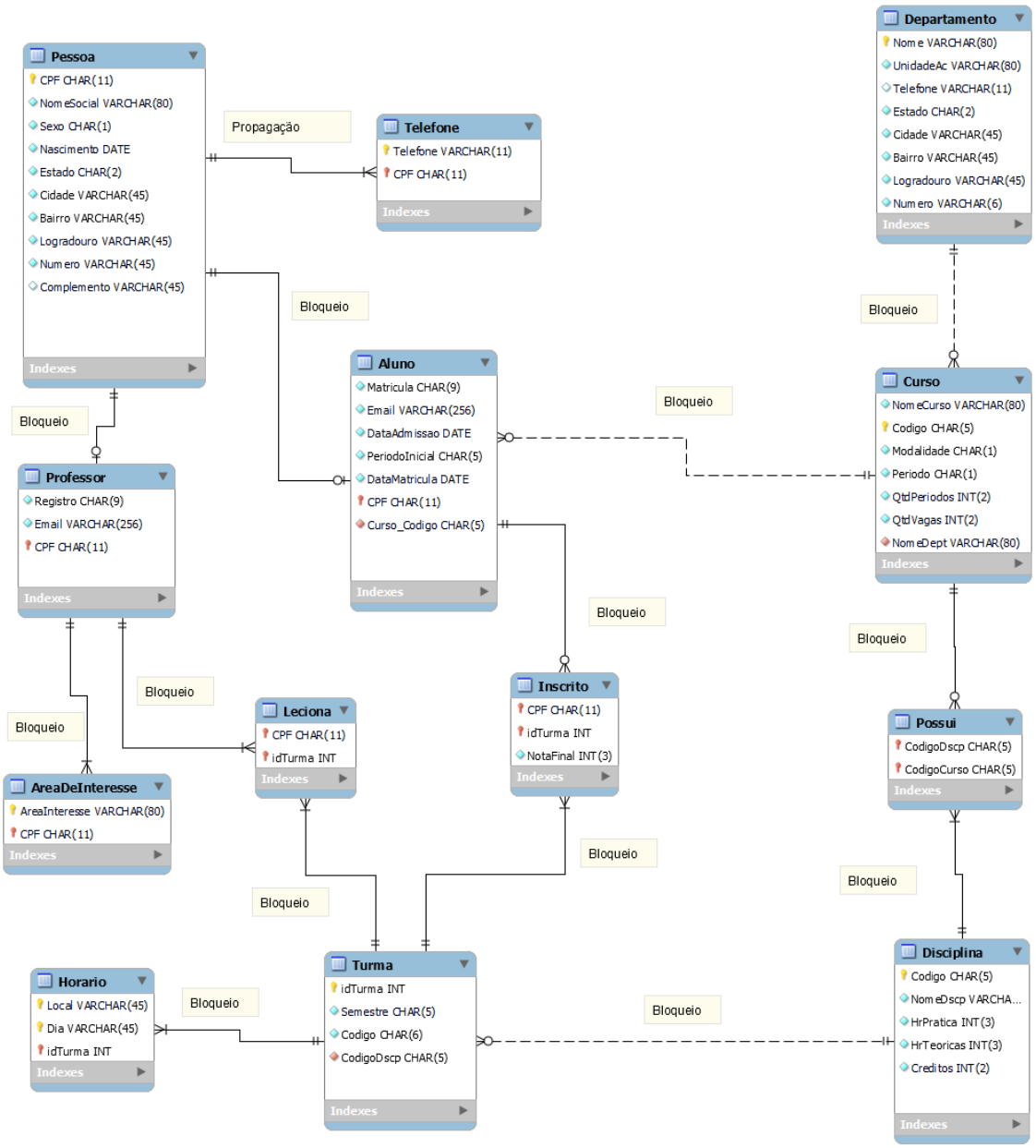
Tipo Relacionamento	Leciona
Descrição	Indica qual turma um professor leciona

Tipo Relacionamento	Tem
Descrição	Indica a qual disciplina uma turma pertence

Tipo Relacionamento	Pertence
Descrição	Indica o departamento que um curso pertence

Tipo Relacionamento	Possui
Descrição	Indica a qual curso uma disciplina pertence

Diagrama Relacional:



Modelo para dicionário de dados:

Tabela	Pessoa
Descrição	Conjunto das pessoas que fazem parte da universidade
Atributos	
Nome	**Descrição**
CPF	Cadastro de pessoa física
Nome social	Designação pela qual a pessoa transexual se identifica
Sexo	Gênero da pessoa
Nascimento	Data de nascimento
Estado	Sigla do estado

Cidade	Nome da cidade
Bairro	Nome do bairro
Logradouro	Nome do logradouro
Número	Número da residência
Complemento	Número do apartamento

Tabela	Telefone
Descrição	Conjunto de telefones relacionados ao CPF
Atributos	
Nome	**Descrição**
Telefone	Número (s) de telefone do usuário
CPF	Cadastro de pessoa física

Tabela	Professor
Descrição	Conjuntos de professores matriculados no sistema acadêmico
Atributos	
Nome	**Descrição**
Registro	Registro do professor
E-mail	E-mail institucional
CPF	Cadastro de pessoa física

Tabela	Área de interesse
Descrição	Áreas de interesse de uma Pessoa
Atributos	
Nome	**Descrição**
Área de interesse	Área em que o professor atua
CPF	Cadastro de pessoa física

Tabela	Leciona
Descrição	Indica em qual turma o aluno está
Atributos	
Nome	**Descrição**
CPF	Cadastro de pessoa física
Id turma	Número identificador da turma

Tabela	Turma
Descrição	Conjunto de turmas das disciplinas
Atributos	
Nome	**Descrição**
Id turma	Número identificador da turma
Semestre	Semestre em que a turma é ofertada
Código	Código de identificação da disciplina
Código disciplina	Código da turma em questão

Tabela	Horário
Descrição	Conjunto de dados que define em que dia e hora da semana que uma aula é realizada
Atributos	

Nome	Descrição
Local	Sala e pavilhão em que uma matéria será realizada (Ex: DCC01, PV2-301, etc.)
Dia	Dia da semana em que a aula será realizada e horário de inicio
Id turma	Número identificador da turma

Tabela	Aluno
Descrição	Conjuntos de alunos matriculados no sistema acadêmico
Atributos	
Nome	**Descrição**
Matrícula	Número de matrícula do aluno
E-mail	E-mail institucional
Data admissão	Data em que o aluno se matriculou
Período inicial	Data do primeiro período do usuário
CPF	Cadastro de pessoa física
Curso código	Código de identificação do curso

Tabela	Inscrito
Descrição	Indica em qual turma o aluno está
Atributos	
Nome	**Descrição**
CPF	Cadastro de pessoa física
Id turma	Número identificador da turma
Nota final	Nota final nas disciplinas do período

Tabela	Curso
Descrição	Cursos que são ministrados na faculdade
Atributos	
Nome	**Descrição**
Nome curso	Nome do curso do usuário
Código	Código de identificação do curso
Modalidade	Especificação do curso (Bacharelado (b) / Licenciatura (L))
Período	Período no dia em que o curso é ministrado (Integral (I) / Matutino (M) / Diurno (E) / Noturno (N))
Qnt períodos	Número de períodos por curso
Qtd vagas	Número de vagas ofertadas no curso
Nome dept	Nome do departamento do curso

Tabela	Possui
Descrição	Indica a qual curso uma disciplina pertence
Atributos	
Nome	**Descrição**
Código disciplina	Código da turma em questão
Código curso	Código de identificação da disciplina

Tabela	Disciplina
Descrição	Disciplinas que são ofertadas pelos cursos de graduação
Atributos	
Nome	**Descrição**

Código	Código de identificação do curso
Nome Discp	Nome da disciplina
Hr prática	Quantidade de horas práticas
Hr teórica	Quantidade de horas teóricas
Créditos	Número de créditos por disciplina

Tabela	Departamento
Descrição	Setor aplicado em determinada área de ensino e pesquisa
Atributos	
Nome	**Descrição**
Nome	Nome do departamento
Unidade Ac	
Telefone	Telefone (s) do departamento
Estado	Sigla do estado
Cidade	Nome da cidade
Bairro	Nome do bairro
Logradouro	Nome do logradouro
Número	Número do prédio

Etapa – 3

a) Criação de todas as tabelas e de todas as restrições de integridade

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Pessoa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pessoa` (
  `CPF` CHAR(11) NOT NULL,
  `NomeSocial` VARCHAR(80) NOT NULL,
  `Sexo` CHAR(1) NOT NULL,
  `Nascimento` DATE NOT NULL,
  `Estado` CHAR(2) NOT NULL,
  `Cidade` VARCHAR(45) NOT NULL,
  `Bairro` VARCHAR(45) NOT NULL,
  `Logradouro` VARCHAR(45) NOT NULL,
  `Numero` VARCHAR(45) NOT NULL,
  `Complemento` VARCHAR(45) NULL,
  PRIMARY KEY (`CPF`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`Telefone`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Telefone` (
  `Telefone` VARCHAR(11) NOT NULL,
  `CPF` CHAR(11) NOT NULL,
  INDEX `fk_Telefone_Pessoa_idx` (`CPF` ASC) VISIBLE,
  PRIMARY KEY (`Telefone`, `CPF`),
  CONSTRAINT `fk_Telefone_Pessoa`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Pessoa` (`CPF`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Professor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Professor` (
  `Registro` CHAR(9) NOT NULL,
  `Email` VARCHAR(256) NOT NULL,
  `CPF` CHAR(11) NOT NULL,
  PRIMARY KEY (`CPF`),
  INDEX `fk_Professor_Pessoa1_idx` (`CPF` ASC) VISIBLE,
  UNIQUE INDEX `Registro_UNIQUE` (`Registro` ASC) VISIBLE,
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE,
  CONSTRAINT `fk_Professor_Pessoa1`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Pessoa` (`CPF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`Departamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Departamento` (
  `Nome` VARCHAR(80) NOT NULL,
  `UnidadeAc` VARCHAR(80) NOT NULL,
  `Telefone` VARCHAR(11) NULL,
  `Estado` CHAR(2) NOT NULL,
  `Cidade` VARCHAR(45) DEFAULT 'Lavras',
  `Bairro` VARCHAR(45) NOT NULL,
  `Logradouro` VARCHAR(45) NOT NULL,
  `Numero` VARCHAR(6) NOT NULL,
  PRIMARY KEY (`Nome`))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Curso`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Curso` (
  `NomeCurso` VARCHAR(80) NOT NULL,
  `Codigo` CHAR(5) NOT NULL,
  `Modalidade` CHAR(1) NOT NULL,
  `Periodo` CHAR(1) NOT NULL,
  `QtdPeriodos` INT(2) NOT NULL,
  `QtdVagas` INT(2) NOT NULL,
  `NomeDept` VARCHAR(80) NOT NULL,
  PRIMARY KEY (`Codigo`),
  INDEX `fk_Curso_Departamento1_idx` (`NomeDept` ASC) VISIBLE,
  CONSTRAINT `fk_Curso_Departamento1`
    FOREIGN KEY (`NomeDept`)
    REFERENCES `mydb`.`Departamento` (`Nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Aluno`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Aluno` (
  `Matricula` CHAR(9) NOT NULL,
  `Email` VARCHAR(256) NOT NULL,
  `DataAdmissao` DATE NOT NULL,
  `PeriodoInicial` CHAR(5) NOT NULL,
  `DataMatricula` DATE NOT NULL,
  `CPF` CHAR(11) NOT NULL,
  `Curso_Codigo` CHAR(5) NOT NULL,
  INDEX `fk_Aluno_Pessoa1_idx` (`CPF` ASC) VISIBLE,
  UNIQUE INDEX `Matricula_UNIQUE` (`Matricula` ASC) VISIBLE,
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE,
  INDEX `fk_Aluno_Curso1_idx` (`Curso_Codigo` ASC) VISIBLE,
  PRIMARY KEY (`CPF`),
  CONSTRAINT `fk_Aluno_Pessoa1`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Pessoa` (`CPF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
   CONSTRAINT `fk_Aluno_Curso1`
     FOREIGN KEY (`Curso_Codigo`)
     REFERENCES `mydb`.`Curso` (`Codigo`)
     ON DELETE NO ACTION
     ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`AreaDeInteresse`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`AreaDeInteresse` (
  `AreaInteresse` VARCHAR(80) NOT NULL,
  `CPF` CHAR(11) NOT NULL,
  PRIMARY KEY (`AreaInteresse`, `CPF`),
  INDEX `fk_AreaDeInteresse_Professor1_idx` (`CPF` ASC) VISIBLE,
  CONSTRAINT `fk_AreaDeInteresse_Professor1`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Professor` (`CPF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`Disciplina`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Disciplina` (
  `Codigo` CHAR(5) NOT NULL,
  `NomeDscp` VARCHAR(80) NOT NULL,
  `HrPratica` INT(3) NOT NULL,
  `HrTeoricas` INT(3) NOT NULL,
  `Creditos` INT(2) NOT NULL,
  PRIMARY KEY (`Codigo`))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Possui`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Possui` (
  `CodigoDscp` CHAR(5) NOT NULL,
  `CodigoCurso` CHAR(5) NOT NULL,
  PRIMARY KEY (`CodigoDscp`, `CodigoCurso`),
  INDEX `fk_Disciplina_has_Curso_Curso1_idx` (`CodigoCurso` ASC) VISIBLE,
  INDEX `fk_Disciplina_has_Curso_Disciplina1_idx` (`CodigoDscp` ASC) VISIBLE,
  CONSTRAINT `fk_Disciplina_has_Curso_Disciplina1`
    FOREIGN KEY (`CodigoDscp`)
    REFERENCES `mydb`.`Disciplina` (`Codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Disciplina_has_Curso_Curso1`
    FOREIGN KEY (`CodigoCurso`)
    REFERENCES `mydb`.`Curso` (`Codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`Turma`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Turma` (
  `idTurma` INT NOT NULL,
  `Semestre` CHAR(5) NOT NULL,
  `Codigo` CHAR(6) NOT NULL,
  `CodigoDscp` CHAR(5) NOT NULL,
  PRIMARY KEY (`idTurma`),
  UNIQUE INDEX `Semestre_UNIQUE` (`Semestre` ASC, `Codigo` ASC, `CodigoDscp` ASC) VISIBLE,
  INDEX `fk_Turma_Disciplina1_idx` (`CodigoDscp` ASC) VISIBLE,
  CONSTRAINT `fk_Turma_Disciplina1`
    FOREIGN KEY (`CodigoDscp`)
    REFERENCES `mydb`.`Disciplina` (`Codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`Horario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Horario` (
  `Local` VARCHAR(45) NOT NULL,
  `Dia` VARCHAR(45) NOT NULL,
  `idTurma` INT NOT NULL,
  PRIMARY KEY (`idTurma`, `Dia`, `Local`),
  CONSTRAINT `fk_Horario_Turma1`
    FOREIGN KEY (`idTurma`)
    REFERENCES `mydb`.`Turma` (`idTurma`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `mydb`.`Leciona`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Leciona` (
  `CPF` CHAR(11) NOT NULL,
  `idTurma` INT NOT NULL,
  PRIMARY KEY (`CPF`, `idTurma`),
  INDEX `fk_Professor_has_Turma_Turma1_idx` (`idTurma` ASC) VISIBLE,
  INDEX `fk_Professor_has_Turma_Professor1_idx` (`CPF` ASC) VISIBLE,
  CONSTRAINT `fk_Professor_has_Turma_Professor1`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Professor` (`CPF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Professor_has_Turma_Turma1`
    FOREIGN KEY (`idTurma`)
    REFERENCES `mydb`.`Turma` (`idTurma`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Inscrito`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Inscrito` (
  `CPF` CHAR(11) NOT NULL,
  `idTurma` INT NOT NULL,
  `NotaFinal` INT(3) NOT NULL,
  PRIMARY KEY (`CPF`, `idTurma`),
  INDEX `fk_Aluno_has_Turma_Turma1_idx` (`idTurma` ASC) VISIBLE,
  INDEX `fk_Aluno_has_Turma_Aluno1_idx` (`CPF` ASC) VISIBLE,
  CONSTRAINT `fk_Aluno_has_Turma_Aluno1`
    FOREIGN KEY (`CPF`)
    REFERENCES `mydb`.`Aluno` (`CPF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Aluno_has_Turma_Turma1`
    FOREIGN KEY (`idTurma`)
    REFERENCES `mydb`.`Turma` (`idTurma`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

b) ALTER TABLE e DROP TABLE:

```
ALTER TABLE `mydb`.`Pessoa`
ADD COLUMN `DataNascimento` DATE NOT NULL FIRST;

ALTER TABLE `mydb`.`Pessoa`
DROP COLUMN `DataNascimento`;

ALTER TABLE `mydb`.`Aluno`
RENAME TO `Discente`;
```

Realiza a alteração na tabela pessoa ao adicionar o atributo DataNascimento

Realiza a alteração na tabela pessoa ao excluir o atributo DataNascimento

Realiza a alteração na tabela aluno ao renomear a mesma para discente

```sql
-- Criando uma tabela fake para a exclusao da mesma
CREATE TABLE IF NOT EXISTS `mydb`.`FakeTable` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(50) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id`)
)
ENGINE = InnoDB;
DROP TABLE `mydb`.`FakeTable`;
```

Criação de uma tabela para realizar a exclusão da mesma

c) Exemplo de inserção de dados em cada uma das tabelas:

```sql
INSERT INTO Pessoa
VALUES ('43210682151','Muhammad Avdol','M','2000-05-19','PR','Curitiba',
'Cidade Industrial','Avenida Juscelino Kubitschek','555',NULL);
```

Insere novos dados na tabela pessoa

```sql
INSERT INTO Telefone
VALUES ('31517252069','06980759000');
```

Insere novos dados na tabela telefone

```sql
INSERT INTO Professor
VALUES ('68282116','costellio@gmail.com','12686571071');
```

Insere novos dados na tabela professor

```sql
INSERT INTO AreaDeInteresse
VALUES ('Banco de dados','12686571071');
```

Insere novos dados na tabela AreaDeInteresse

```sql
INSERT INTO Departamento
VALUES ('Departamento de ciencias exatas','Ciencias exatas','21 23837753',
'MG','Lavras','Av central','Professor Edmir Sá Santos','665');
```

Insere novos dados na tabela departamento

```sql
INSERT INTO Curso
VALUES ('Fisica','67369','L','N','8','40',
'Departamento de ciencias exatas');
```

Insere novos dados na tabela curso

```
INSERT INTO Disciplina
VALUES ('10000','Calculo Numerico','120','120','4');
```

Insere novos dados na tabela disciplina

```
INSERT INTO Turma
VALUES ('1','20221','10A','10000');
```

Insere novos dados na tabela turma

```
INSERT INTO Possui
VALUES ('10000','22046');
```

Insere novos dados na tabela possui

```
INSERT INTO Aluno
VALUES ('989194900','angelo@gmail.com','2020-01-01',
'20201','2020-01-01','06980759000','67369');
```

Insere novos dados na tabela aluno

```
INSERT INTO Inscrito
VALUES ('06980759000','1','100');
```

Insere novos dados na tabela inscrito

```
INSERT INTO Leciona
VALUES ('12686571071','1');
```

Insere novos dados na tabela leciona

```
INSERT INTO Horario
VALUES ('Pavilhão 6','Segunda Feira 10:00','1');
```

Insere novos dados na tabela horário

d) Exemplos de modificação de dados e UPDATE aninhado:

Aninhado:

```sql
UPDATE Aluno
SET Email = 'novo3email@universidade.com'
WHERE CPF IN(
  SELECT CPF
  FROM Pessoa
  WHERE CPF = '71214485219'
);
-- Teste para esse update
SELECT * FROM Aluno
WHERE CPF = (
  SELECT CPF
  FROM Pessoa
  WHERE CPF = '71214485219'
);
```

Altera o e-mail que estava associado a um determinado aluno e o altera para “novo3email@universidade.com”
O update em aluno deve ser feito pelo cpf da tabela pessoa, pois é uma chave estrangeira de aluno

Updates:

```sql
-- Update Departamento
Update Departamento d
Set d.Telefone = '22 52149635'
Where d.Nome = 'Departamento de biologia';
-- Teste
select * FROM departamento;
```

Altera o telefone que estava associado a um determinado departamento e o altera para “22 52149645”

```sql
-- update Curso
UPDATE Curso c
SET c.QtdVagas = c.QtdVagas - 1
WHERE c.Codigo = '12019';
-- Teste
select * from Curso ;
```

subtrai a quantidade de vagas de um curso selecionado por seu código em 1

```sql
-- Update Professor
UPDATE Professor p
INNER JOIN Pessoa pe ON p.CPF = Pe.CPF
SET p.Email = 'novoemail@universidade.com'
WHERE pe.CPF =  '12686571071';
-- Teste
select * from Professor ;
```

Altera o e-mail que estava associado a um determinado cpf e o altera para "novoemail@universidade.com"

```sql
-- Update Pessoa
UPDATE pessoa pe
SET pe.NomeSocial = 'Maria Silva'
WHERE pe.CPF = '90097344397';
-- Teste
SELECT  pe.NomeSocial
FROM pessoa pe
WHERE pe.CPF = '90097344397';
```

Altera o nome social que estava associado a um determinado cpf e o altera para "Maria Silva"

e) Exemplos de exclusão de dados e DELETE aninhado:

Dados de teste:

```sql
-- Exemplo de delete
INSERT INTO Departamento
VALUES ('Departamento de exclusão','Exemplo de exclusão','00 00000000','NL','Exemplo','Exemplo','Exemplo','00')
INSERT INTO Curso
VALUES ('Exclusão','00000','L','N','0','0','Departamento de exclusão')
```

Delete aninhado:

```sql
DELETE FROM Curso
WHERE NomeDept IN (select Nome from Departamento WHERE Nome = 'Departamento de exclusão');
```

Deleta todas as linhas da tabela curso cujo NomeDept é igual a "departamento de exclusão"

Exclusões:

```sql
-- Delete Testee 2/5

DELETE FROM Departamento
WHERE nome IN Departamento = 'Departamento de exclusão'
```

Deleta a tupla cujo atribuo nome é igual a departamento de exclusão

```sql
-- Delete pelo CPF 3/5
INSERT INTO Telefone
VALUES ('19 95682705','06980759000')

DELETE FROM Telefone
WHERE CPF IN Telefone = '06980759000';
```

Deleta a tupla cujo cpf apresente o valor indicado

```sql
-- Delete pelo Numero 4/5
INSERT INTO Telefone
VALUES ('19 95682705','06980759000')

DELETE FROM Telefone
WHERE Telefone IN Telefone = '19 95682705';
```

Deleta a tupla cujo telefone apresenta o valor indicado

```sql
-- exclui todos elementos em Teste
-- no MySQL, o modo "safe update" deve ser desabilitado
-- Edit --> Preferences --> SQL Editor --> desmarque "Safe Updates"
-- Só passa a valer a partir da próxima conexão
CREATE TABLE IF NOT EXISTS `mydb`.`Teste` (
  `Nome` VARCHAR(30) NOT NULL,
  `ID`   INT(2) NOT NULL,
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

INSERT INTO Teste
VALUES ('Nome Teste 1','0')
INSERT INTO Teste
VALUES ('Nome Teste 2','1')
INSERT INTO Teste
VALUES ('Nome Teste 3','2')

DELETE FROM Teste

DROP TABLE Teste
```

f) Exemplos de, pelo menos, 12 consultas:

```sql
-- Retorna nome da disciplina e código da turma que a professora "Lisa Lisa" leciona.
SELECT NomeDscp, T.Codigo FROM disciplina D, pessoa N, professor P, turma T , leciona L
WHERE N.NomeSocial = 'Lisa Lisa' AND N.CPF = P.CPF AND  P.CPF = L.CPF AND T.idTurma = L.idTurma AND T.CodigoDscp = D.Codigo;

-- Em que departamento cada professor trabalha.
SELECT DISTINCT NomeSocial, Nome FROM departamento D, curso C, possui P, disciplina DI, turma T, leciona L, professor PR, pessoa PE
WHERE D.Nome = C.NomeDept AND C.Codigo = P.CodigoCurso AND P.CodigoDscp = DI.Codigo
AND DI.Codigo = T.CodigoDscp AND T.IdTurma = L.idTurma AND L.CPF = PR.CPF AND PR.CPF = PE.CPF;

-- quantas turmas cada disciplina tem.
SELECT NomeDscp, COUNT(*) AS qntdTurmas FROM turma T, disciplina D
WHERE T.CodigoDscp = D.Codigo GROUP BY NomeDscp;

-- Alunos com CRA maior que 60,0
SELECT NomeSocial, AVG(NotaFinal) AS CRA FROM pessoa P, aluno A, inscrito I
WHERE P.CPF = A.CPF AND A.CPF = I.CPF GROUP BY NomeSocial HAVING CRA >= 60;

-- Alunos que possuem nome que comece com a letra 'J'
SELECT NomeSocial FROM pessoa NATURAL JOIN aluno WHERE NomeSocial LIKE 'J%';


-- Alunos que possuem nome que comece com a letra 'J'
SELECT NomeSocial FROM pessoa NATURAL JOIN aluno WHERE NomeSocial LIKE 'J%';

-- Professores que não possuem complemento
SELECT NomeSocial FROM pessoa NATURAL JOIN professor WHERE Complemento IS NULL;

-- Alunos de Fisica que tem a nota maior que algum aluno de Matematica.
SELECT NomeSocial, NotaFinal FROM pessoa P, aluno A, curso C, inscrito I
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND A.Curso_Codigo = C.Codigo AND C.NomeCurso = 'Fisica' AND I.NotaFinal > SOME
(SELECT NotaFinal FROM pessoa P, aluno A, curso C, inscrito I
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND A.Curso_Codigo = C.Codigo AND C.NomeCurso = 'Matematica');

-- Alunos de Fisica que tem a nota maior que todos os alunos de Matematica.
SELECT NomeSocial, NotaFinal FROM pessoa P, aluno A, curso C, inscrito I
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND A.Curso_Codigo = C.Codigo AND C.NomeCurso = 'Fisica' AND I.NotaFinal > ALL
(SELECT NotaFinal FROM pessoa P, aluno A, curso C, inscrito I
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND A.Curso_Codigo = C.Codigo AND C.NomeCurso = 'Matematica');
```

```sql
-- Disciplinas que cada curso tem.
SELECT NomeCurso, NomeDscp FROM curso C, possui P, disciplina D
WHERE C.Codigo = P.CodigoCurso AND P.CodigoDscp = D.Codigo;

-- Alunos de cada curso e datas de matricul.a
SELECT NomeCurso, NomeSocial, DataMatricula FROM  pessoa P, aluno A, curso C
WHERE P.CPF = A.CPF AND A.Curso_Codigo = C.codigo ;

-- Alunos que se matricularam entre 2019 e 2020
SELECT NomeSocial, DataMatricula FROM pessoa P, aluno A
WHERE P.CPF = A.CPF AND (DataMatricula BETWEEN '2019-01-01' AND '2020-01-01');

-- se tiber turma na terça, retorne quais são essas turmas.
SELECT T.idTurma, Codigo FROM turma T, horario H
WHERE T.idturma = H.idTurma AND H.Dia LIKE 'Terça%' AND EXISTS
(SELECT * FROM horario
WHERE Dia LIKE 'Terça%');

-- Recupera o Nome e a Nota Final ordena de forma Descrecente dos Alunos aprovados em Algoritmos em Grafos
 SELECT NomeSocial, NotaFinal FROM pessoa P, aluno A , turma T, inscrito I, disciplina D
 WHERE P.CPF = A.CPF
 AND I.NotaFinal > 60 AND A.CPF = I.CPF AND I.idTurma = T.idTurma AND D.NomeDscp = 'Algoritimos em Grafos'
 AND D.Codigo = T.CodigoDscp ORDER BY NotaFinal DESC;

 -- Recupera o Nome da Pessoa e seu telefone caso não possua retorna Nulo
 Select NomeSocial, Telefone FROM pessoa P LEFT OUTER JOIN Telefone T ON P.CPF = T.CPF;

 -- Recupera o nome dos alunos que iniciaaram o curso no periodo 2020-1 ou 2021-2
SELECT NomeSocial FROM aluno JOIN pessoa ON aluno.CPF = pessoa.CPF
WHERE aluno.PeriodoInicial = '20201' or aluno.PeriodoInicial = '20212';

-- Retorna os departamentos que não tem curso associado a eles
 SELECT Nome FROM departamento WHERE Nome NOT IN (SELECT NomeDept FROM curso);

-- Retorna o nome do curso que pertence ao departamento de computação ou cursos que possuem pelo menos 10 períodos
 SELECT NomeCurso FROM curso JOIN departamento ON  departamento.Nome = curso.NomeDept
 WHERE NomeDept = 'Departamento de Ciência da Computação'
 UNION SELECT NomeCurso FROM curso WHERE QtdPeriodos >= '10';

 -- Recupera o nome dos alunos que iniciaaram o curso no periodo 2020-1 ou 2021-2
SELECT NomeSocial FROM aluno JOIN pessoa ON aluno.CPF = pessoa.CPF
WHERE aluno.PeriodoInicial = '20201' OR aluno.PeriodoInicial = '20212';
```

g) Exemplos de criação de de 3 visões:

```sql
-- Cria uma visão que retorna uma matéria e o nome do professor que a leciona
CREATE VIEW docenteMateria AS SELECT DISTINCT nomeDscp, NomeSocial FROM disciplina D, pessoa P, professor PE, leciona L, turma T
WHERE P.CPF = PE.CPF AND PE.CPF = L.CPF AND L.idTurma = T.idTurma AND T.CodigoDscp = D.Codigo;

-- Cria uma visão com o departamento e  os cursos que ele possui
CREATE VIEW CursoDepartamento AS SELECT Nome, NomeCurso
FROM departamento JOIN curso ON departamento.Nome = curso.NomeDept;
```

```sql
-- Cria uma visão que retorna turma e disciplina e o semestre
CREATE VIEW DisciplinaTurma AS SELECT NomeDscp, idTurma, Semestre
FROM turma JOIN disciplina ON turma.CodigoDscp = disciplina.Codigo;

-- Conta todas as turmas em DisciplinaTurma
SELECT count(NomeDscp) FROM DisciplinaTurma;
```

h) Exemplos de criação de usuários, concessão e revogação de permissão de acesso:

Criação:

```sql
CREATE USER 'heitor'@'localhost' IDENTIFIED BY '090507';

CREATE USER 'lucas'@'localhost' IDENTIFIED BY '887213';

CREATE USER 'bryan'@'localhost'IDENTIFIED BY '1567890';
```

Concessão:

```sql
-- Concede permissão de update para o usuário Bryan nos atributos HrPratica e HrTeoricas na tabela Disciplina
GRANT UPDATE (HrPratica, HrTeoricas) ON mydb.disciplina TO 'bryan'@'localhost';

-- Concede permissão de select e update para o usuário Heitor em todos os atributos da tabela departamento
GRANT SELECT, UPDATE ON mydb.departamento TO 'heitor'@'localhost';
```

Revogação:

```sql
-- Revoga a permissão de update dos atributos HrPratica e HrTeoricas da tabela Disciplina
REVOKE UPDATE (HrPratica, HrTeoricas) ON mydb.disciplina FROM 'bryan'@'localhost';

-- Revoga a permissão de exclusão da tabela pessoa do usuário Heitor
REVOKE DELETE ON mydb.pessoa FROM 'heitor'@'localhost';
```

i) Exemplos de procedimentos/funções:

```sql
-- STORE PROCEDURES

use mydb;

-- aluno aleatorio de uma disciplina
DELIMITER //
drop procedure if exists randomStudent //
CREATE PROCEDURE randomStudent (IN pDisciplina VARCHAR(80), OUT randomStudentName VARCHAR (80))
BEGIN
    SELECT NomeSocial INTO randomStudentName
    FROM pessoa P, aluno A, curso C, possui PO, disciplina D
    WHERE P.CPF = A.CPF AND A.Curso_Codigo = C.Codigo AND C.Codigo = PO.CodigoCurso AND PO.CodigoDscp = D.Codigo ORDER BY RAND()
    LIMIT 1;
END //
DELIMITER ;

-- Testando store procedure

CALL randomStudent ('Calculo I', @NomeAluno);
SELECT @NomeAluno AS alunoAleatorio;
```

```sql
-- -----------------------------------------------------------

/*
sistema de balanceamento de notas. Ex: Um professor viu que aplicou uma prova complexa demais
e decidiu aumentar as notas de forma que ficasse justo para todos. Sendo assim, ele viu que era
melhor aumentar a nota dos que tiraram menos de 30% em 25% há mais da nota tirada e dos que tiraram
entre 31% e 59%, ele aumentou em 10%. Com o fim de equilibrar as notas.
*/

DELIMITER //
drop procedure if exists pontoExtra //
CREATE PROCEDURE pontoExtra (IN pNomeDscp VARCHAR(80))
BEGIN
    DECLARE done INT DEFAULT FALSE;

    DECLARE vCPF CHAR(11);
    DECLARE vNomeSocial VARCHAR(80);
    DECLARE vNotaFinal INT;

   DECLARE extraPoint CURSOR FOR
      SELECT I.CPF, P.NomeSocial, I.NotaFinal
      FROM pessoa P, aluno A, inscrito I, turma T, disciplina D
      WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND I.idTurma = T.idTurma AND T.CodigoDscp = D.Codigo AND D.NomeDscp = pNomeDscp;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN extraPoint;

   read_loop: LOOP
      FETCH extraPoint INTO vCPF, vNomeSocial, vNotaFinal;

      IF done THEN
          LEAVE read_loop;
      END IF;

       IF vNotaFinal > 30 AND vNotaFinal < 60 THEN
           UPDATE inscrito SET NotaFinal = NotaFinal * 1.1 WHERE CPF = vCPF;
       ELSEIF vNotaFinal > 0 AND vNotaFinal <= 30 THEN
           UPDATE inscrito SET NotaFinal = NotaFinal * 1.25 WHERE CPF = vCPF;
       END IF;

    END LOOP;

    CLOSE extraPoint;

END //
DELIMITER ;
/*
Testando o store procedure, ao chamar o procedimento, ele balanceará as notas que foram abaixo da
média em uma disciplina específica
*/

SELECT I.CPF, NomeSocial, NotaFinal FROM pessoa P, aluno A, inscrito I, turma T, disciplina D
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND I.idTurma = T.idTurma AND T.CodigoDscp = D.Codigo AND D.NomeDscp = 'Arquitetura de computadores';

CALL pontoExtra ('Arquitetura de computadores');

SELECT I.CPF, NomeSocial, NotaFinal FROM pessoa P, aluno A, inscrito I, turma T, disciplina D
WHERE P.CPF = A.CPF AND A.CPF = I.CPF AND I.idTurma = T.idTurma AND T.CodigoDscp = D.Codigo AND D.NomeDscp = 'Arquitetura de computadores';


-- -----------------------------------------------------------

/*
Capitaliza todos os nomes na coluna `NomeSocial` na tabela 'pessoa'.
*/
```

```
DELIMITER //
drop procedure if exists lowerNames //
CREATE PROCEDURE lowerNames ()
BEGIN
    DECLARE done INT DEFAULT FALSE;

    DECLARE vCPF CHAR (11);
    DECLARE vNomeSocial VARCHAR(80);

    DECLARE i INT;
    DECLARE c, sc CHAR (1);
    DECLARE outstr VARCHAR(1000);

    DECLARE lowering CURSOR FOR
        SELECT CPF, NomeSocial
        FROM pessoa;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN lowering;
     read_loop: LOOP
         FETCH lowering INTO vCPF, vNomeSocial;

         IF done THEN
             LEAVE read_loop;
         END IF;

         SET i = 1;
         SET outstr = vNomeSocial;
         WHILE i<= CHAR_LENGTH(vNomeSocial) DO
             SET c = SUBSTRING(vNomeSocial, i, 1);
             SET sc = CASE WHEN i = 1 THEN ' '
                       ELSE SUBSTRING(vNomeSocial, i - 1, 1)
                       END;
             IF sc IN (' ') THEN
                 SET outstr = INSERT(outstr, i, 1, UPPER(c));
             END IF;
             SET i = i+1;
         END WHILE;
         SET vNomeSocial = outstr;

         WHILE i<= CHAR_LENGTH(vNomeSocial) DO
             SET c = SUBSTRING(vNomeSocial, i, 1);
             SET sc = CASE WHEN i = 1 THEN ' '
                       ELSE SUBSTRING(vNomeSocial, i - 1, 1)
                       END;
             IF sc IN (' ') THEN
                 SET outstr = INSERT(outstr, i, 1, UPPER(c));
             END IF;
             SET i = i+1;
         END WHILE;
         SET vNomeSocial = outstr;

         UPDATE pessoa SET NomeSocial = vNomeSocial WHERE CPF = vCPF;

    END LOOP;

    CLOSE lowering;

END //
DELIMITER ;
```

```sql
-- Testando o store procedure, capitalizando o nome da pessoa 'manoel gomes'.

SELECT NomeSocial FROM pessoa WHERE NomeSocial = 'manoel gomes';

INSERT INTO pessoa
VALUE ('12345678911', 'manoel gomes', 'M', '1969-12-2', 'MA', 'Balsas', 'Centro', 'Rua dos Pássaros', '13', 'Casa 13A');

CALL lowerNames();

SELECT NomeSocial FROM pessoa;
```

j) Exemplos de 3 triggers:

```sql
-- TRIGGERS --
use mydb;

-- trigger de auditoria (UPDATE)
CREATE TABLE auditoriaNotaFinal (
idAuditoria INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
CPFaluno CHAR(11) NOT NULL,
notaAnterior INT (3) NOT NULL,
notaNova INT(3) NOT NULL,
user VARCHAR(20) NOT NULL,
dataHora datetime NOT NULL
);

DELIMITER $$
CREATE TRIGGER after_Professor_update
AFTER UPDATE ON inscrito
FOR EACH ROW
BEGIN
    IF OLD.NotaFinal != NEW.NotaFinal THEN
    INSERT INTO auditoriaNotaFinal (CPFaluno, notaAnterior, notaNova, user, dataHora)
    VALUE (NEW.CPF, OLD.NotaFinal, NEW.NotaFinal, USER(), NOW());
```

```
    END IF;
END $$
DELIMITER ;

--  exemplo aplicando o trigger de auditoria

SELECT * FROM inscrito;

UPDATE inscrito
SET NotaFinal = 60
WHERE CPF = '06980759000' AND idTurma = 2;

SELECT * FROM inscrito;

SELECT * FROM auditoriaNotaFinal;

-- ------------------------------------------------------------

-- Não é possível inserir um curso que contenha mais de 12 periodos (INSERT)
DELIMITER $$
CREATE TRIGGER before_curso_insert
BEFORE INSERT ON curso
FOR EACH ROW
BEGIN
    IF NEW.QtdPeriodos > 12 THEN
    SIGNAL SQLSTATE '45000' SET message_text = 'A quantidade máxima de períodos que um curso pode ter na UFL
    END IF;
END $$
DELIMITER ;

-- Teste do trigger, inserindo um curso com 13 periodos, para disparar o trigger

INSERT INTO curso (NomeCurso, Codigo, Modalidade, Periodo, QtdPeriodos, QtdVagas, NomeDept)
VALUE ('Engenharia de Automação', '7777', 'B', 'I', 13 , 40, 'Departamento de ciencias exatas');


-- ------------------------------------------------------------

-- TRIGGER de horarios diponiveis após uma turma excluir um determinado horário (DELETE)

CREATE TABLE HorariosDispo (
idHorario INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
LocalHor VARCHAR(45) NOT NULL,
Dia VARCHAR(45) NOT NULL
);

DELIMITER $$
CREATE TRIGGER after_horario_delete
AFTER DELETE ON horario
FOR EACH ROW
BEGIN
    INSERT INTO HorariosDispo (LocalHor, Dia)
    VALUE (OLD.Local, OLD.Dia);
END $$
DELIMITER ;

-- Testando trigger, deletando um horario para disparar o trigger e adicionar este horario deletado na tabela de horarios diponiveis

SELECT * from horario;
```

```sql
DELETE FROM Horario WHERE Local = 'Pavilhão 6' AND Dia = 'Segunda Feira 10:00' AND idTurma = 1;

SELECT * from HorariosDispo;
```