DATABASES COURSE

MANDATORY I

Student Group 11

Dorin Moldovan 10.09.1999

Table of Contents

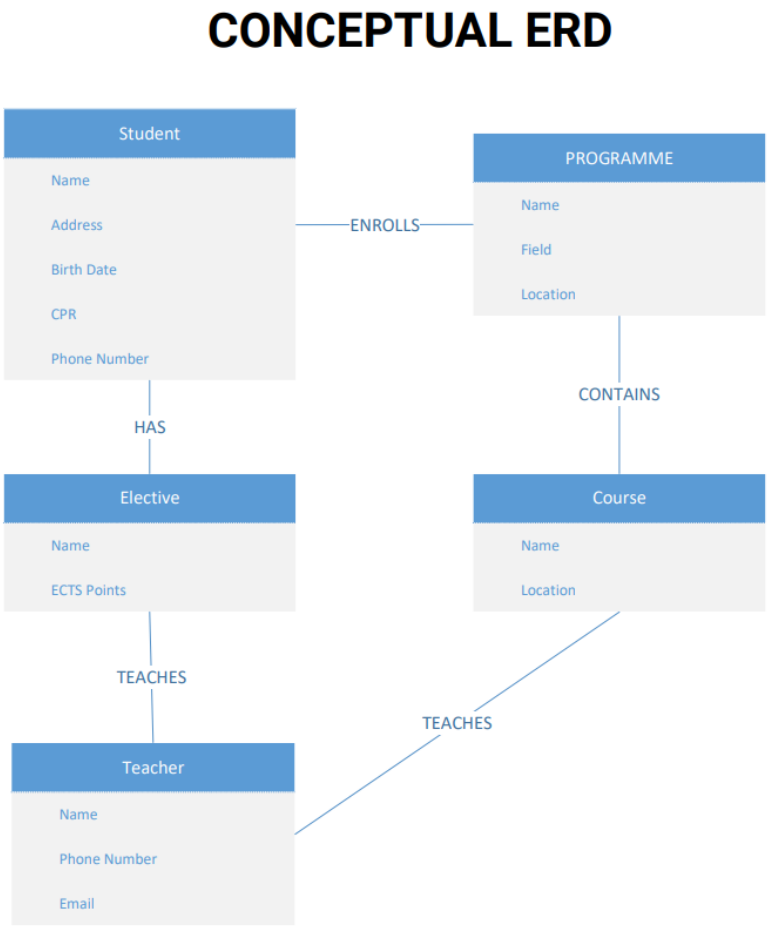
Problem Statement

As I was in a small group, my idea was to create a database with fewer tables to eliminate the redundant procedures, but to include all types of relations (one to one, one to many, many to many) to show that I am able to create a backend that would facilitate the needs of a database with all the relations, especially the many to many one, as it has been tricky for me in the past. For this I chose to create a database for a university, because I don't have any imagination and it seemed to be straighforward.

Model Creation

Conceptual Model

Started by creating the tables for the Student, Programme, Course, Teacher and Elective added some fields and some relations. Was not sure about the direction of the relations though.



Logical Model

Moving to the Logical ERD, added the Primary and Foreign Keys, as well as updated the fields for the tables. Made the relationships using the crowfoot notation. The relationships were made by following those rules set previously by me:

A student can be enrolled in one and only one Programme.

A student can have many or no electives

A Programme can have 0 or many students

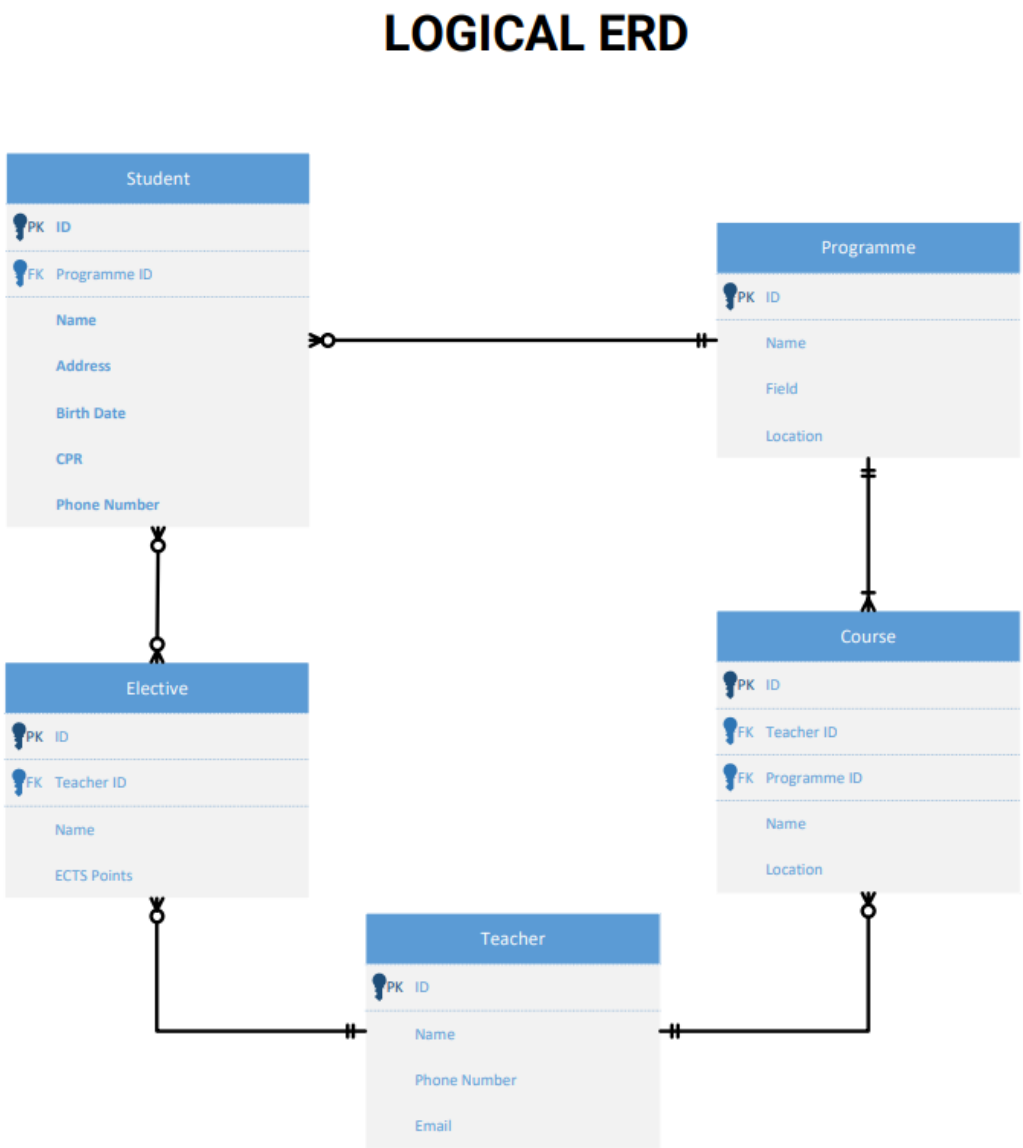
A programme can have 1 or many courses

A Course can be part of one and only one Programme

A Course can have one and only one teacher

A tacher can teach 0 or many courses

A teacher can teach 0 or many electives



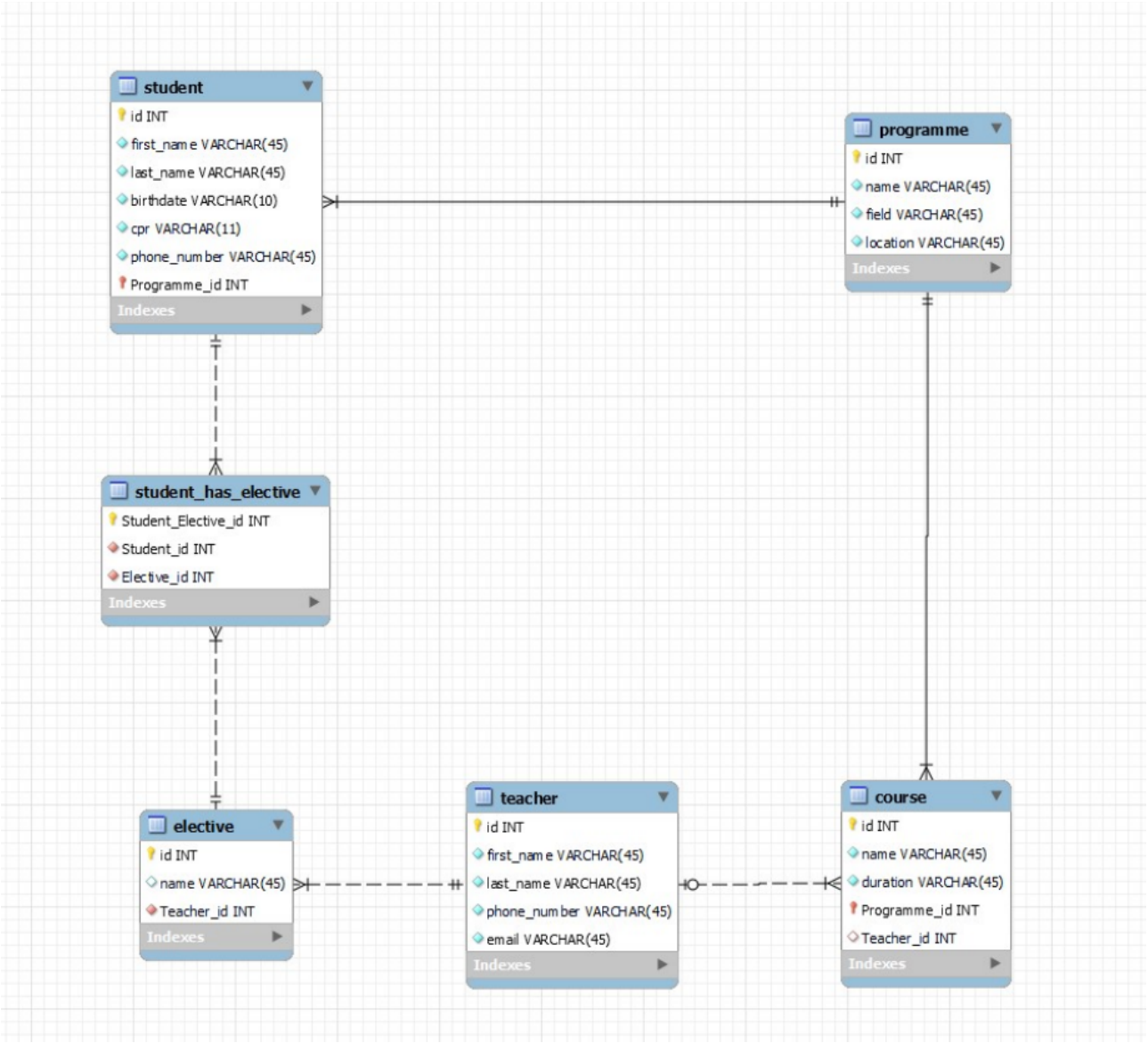
Physical Model

For the Phyisical Model, I described the specific implimentation of the Logical Model, added the junction table for the student to elective many to many relationship. All the fields got their specific types to be as realistic as possible.

There are two identifying relationships:

Student to Programme, as a student cannot exist unless he is in a program.

Course to Programme, for the same reason.



Data Definition

```sql
DROP SCHEMA IF EXISTS `university` ;

CREATE SCHEMA IF NOT EXISTS `university` DEFAULT CHARACTER SET utf8 ;
USE `university` ;

CREATE TABLE IF NOT EXISTS `university`.`programme` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `field` VARCHAR(45) NOT NULL,
  `location` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
AUTO_INCREMENT = 8;


CREATE TABLE IF NOT EXISTS `university`.`teacher` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC));


CREATE TABLE IF NOT EXISTS `university`.`course` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `duration` VARCHAR(45) NOT NULL,
  `Programme_id` INT NOT NULL,
  `Teacher_id` INT NULL,
  PRIMARY KEY (`id`, `Programme_id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_Course_Programme1_idx` (`Programme_id` ASC),
  INDEX `fk_Course_Teacher1_idx` (`Teacher_id` ASC),
  CONSTRAINT `fk_Course_Programme1`
    FOREIGN KEY (`Programme_id`)
    REFERENCES `university`.`programme` (`id`)
    ON DELETE RESTRICT,
  CONSTRAINT `fk_Course_Teacher1`
    FOREIGN KEY (`Teacher_id`)
    REFERENCES `university`.`teacher` (`id`));
```

```sql
CREATE TABLE IF NOT EXISTS `university`.`elective` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NULL DEFAULT NULL,
  `Teacher_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_Elective_Teacher1_idx` (`Teacher_id` ASC),
  CONSTRAINT `fk_Elective_Teacher1`
    FOREIGN KEY (`Teacher_id`)
    REFERENCES `university`.`teacher` (`id`));

CREATE TABLE IF NOT EXISTS `university`.`student` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `birthdate` VARCHAR(10) NOT NULL,
  `cpr` VARCHAR(11) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `Programme_id` INT NOT NULL,
  PRIMARY KEY (`id`, `Programme_id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_Student_Programme_idx` (`Programme_id` ASC),
  CONSTRAINT `fk_Student_Programme`
    FOREIGN KEY (`Programme_id`)
    REFERENCES `university`.`programme` (`id`)
    ON DELETE RESTRICT)
AUTO_INCREMENT = 17;

CREATE TABLE IF NOT EXISTS `university`.`student_has_elective` (
  `Student_Elective_id` INT NOT NULL AUTO_INCREMENT,
  `Student_id` INT NOT NULL,
  `Elective_id` INT NOT NULL,
  INDEX `fk_Student_has_Elective_Student1_idx` (`Student_id` ASC),
  INDEX `fk_Student_has_Elective_Elective1_idx` (`Elective_id` ASC),
  PRIMARY KEY (`Student_Elective_id`),
  CONSTRAINT `fk_Student_has_Elective_Elective1`
    FOREIGN KEY (`Elective_id`)
    REFERENCES `university`.`elective` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Student_has_Elective_Student1`
    FOREIGN KEY (`Student_id`)
    REFERENCES `university`.`student` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Data Manipulation

I came up with some random names, programmes and courses and inserted them into the database.

```sql
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('1', 'Computer Science', 'IT', 'Central');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('2', 'Artificial Intelligence', 'IT', 'Central');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('3', 'Economy', 'ECON', 'North');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('4', 'Finance', 'ECON', 'North');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('5', 'Banking', 'ECON', 'North');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('6', 'International Law', 'LAW', 'West');
INSERT INTO `university`.`programme` (`id`, `name`, `field`, `location`) VALUES ('7', 'Danish Law', 'LAW', 'West');

INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`, `birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('1','John', 'West',
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`, `birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('2','Nigel', 'Bensm
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`,`birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('3','Sam', 'White',
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`,  `birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('4','George', 'Smi
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`,`birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('5','Alex', 'McFord'
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`,`birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('6','Johanne', 'Hans
INSERT INTO `university`.`student` (`id`,`first_name`,`last_name`,`birthdate`, `cpr`, `phone_number`, `Programme_id`) VALUES ('7','Laura', 'Siemer

INSERT INTO `university`.`course` (`id`, `name`, `duration`, `Programme_id`) VALUES ('1', 'Programming 1', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Databases 1', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Programming 2', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Intro to AI ', '3', '2');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Applied AI', '6', '2');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Big Data', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Applied Mathematics', '3', '2');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Networking ', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Discrete Structures', '6', '2');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Algorithms & Data Structures', '6', '1');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Microeconomics', '6', '3');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Macroeconomics', '6', '3');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Applied Statistics', '6', '3');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Intro to Finance', '3', '3');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Maths for Finance', '6', '4');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Excel Spreadsheets', '12', '4');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Big Talk', '6', '4');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Professional Word', '6', '4');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Powerpoint for Meetings', '6', '4');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Intro Banking ', '12', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Economics & Finance', '6', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Suits & Haircuts', '12', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Money Counting', '3', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Money Keeping', '3', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Selling and Buying ', '6', '5');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('EU Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Asia Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('USA Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Intro to Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Law for Finance', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Medical Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Russian Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('South American Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Chinese Law', '6', '6');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Intro to Law', '3', '7');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Housing Law', '3', '7');
INSERT INTO `university`.`course` (`name`, `duration`, `Programme_id`) VALUES ('Business Law', '6', '7');
```

The full script can be seen on the GitHub page of the repository.

Events

For the events I thought of adding the functionality of deleting a student after he graduated, but because I did not think about it before, I had to alter the student table by adding a graduation date field.

```sql
ALTER TABLE `university`.`student`
ADD COLUMN `graduation` DATE NOT NULL AFTER `Programme_id`;

UPDATE `university`.`student` SET `graduation` = '2022-01-01' WHERE (`id` = '1') and (`Programme_id` = '1');
UPDATE `university`.`student` SET `graduation` = '2022-01-01' WHERE (`id` = '2') and (`Programme_id` = '1');
UPDATE `university`.`student` SET `graduation` = '2022-01-01' WHERE (`id` = '3') and (`Programme_id` = '2');
UPDATE `university`.`student` SET `graduation` = '2023-01-01' WHERE (`id` = '4') and (`Programme_id` = '2');
UPDATE `university`.`student` SET `graduation` = '2023-01-01' WHERE (`id` = '5') and (`Programme_id` = '3');
UPDATE `university`.`student` SET `graduation` = '2024-01-01' WHERE (`id` = '6') and (`Programme_id` = '3');
UPDATE `university`.`student` SET `graduation` = '2024-01-01' WHERE (`id` = '7') and (`Programme_id` = '4');

DROP EVENT IF EXISTS student_graduation;
CREATE EVENT student_graduation
    ON schedule
        every 1 year starts '2022-01-01'
    DO
        DELETE FROM student
        WHERE graduation < CURDATE();
```

Triggers

For triggers I couldn't come with a better idea than to trigger a count for how many students, courses, teachers and electives the university has, but because I was not sure where to store this data, I thought it would be a "great" idea to have another, unconnected table in the database, honestly I am not sure about it.

```sql
DROP TRIGGER IF EXISTS new_student;
CREATE TRIGGER new_student
BEFORE INSERT ON student
FOR EACH ROW
UPDATE totals
SET totals.amount = amount + 1 where id = '1';

DROP TRIGGER IF EXISTS student_leaving;
CREATE TRIGGER student_leaving
BEFORE DELETE ON student
FOR EACH ROW
UPDATE totals
SET totals.amount = amount - 1 where id = '1';
```

But basically whenever a new student or course is created or deleted, the count is also updated.

The additional scripts for this to work properly:

```sql
CREATE TABLE `university`.`totals` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `amount` INT NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`));


INSERT INTO `university`.`totals` (`id`, `name`) VALUES ('1', 'students');
INSERT INTO `university`.`totals` (`id`, `name`) VALUES ('2', 'programmes');
INSERT INTO `university`.`totals` (`id`, `name`) VALUES ('3', 'courses');
INSERT INTO `university`.`totals` (`id`, `name`) VALUES ('4', 'electives');
INSERT INTO `university`.`totals` (`id`, `name`) VALUES ('5', 'teachers');
```

Views

I came up with the idea of two views, one for the students and the programme they are enrolled in, and one for the teachers and the courses they teach.

```sql
CREATE VIEW `student_enrollment` AS
    SELECT
        `s`.`id` AS `id`,
        `s`.`first_name` AS `first_name`,
        `s`.`last_name` AS `last_name`,
        `s`.`phone_number` AS `phone_number`,
        `s`.`cpr` AS `cpr`,
        `p`.`name` AS `programme`
    FROM
        (`student` `s`
        JOIN `programme` `p` ON ((`s`.`id` = `p`.`id`)))
```

```sql
SELECT * FROM university.student_enrollment;
```

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

id	first_name	last_name	phone_number	cpr	programme
1	John	West	+45 25684722	220897-4523	Computer Science
2	Nigel	Bensmith	+45 796224100	090598-7421	Artificial Intelligence
3	Sam	White	+45 20566374	140897-1979	Economy
4	George	Smith	+45 63225570	110198-7103	Finance
5	Alex	McFord	+45 42322571	010700-6311	Banking
6	Johanne	Hansen	+45 76129533	161098-9524	International Law
7	Laura	Siemens	+45 50098841	271197-3198	Danish Law

Indexes

Indexed the programme name and the course name because I think most of the queries would be to related to programme names and elective names so it would be nice for them to be retrieved faster, but will probably change or update that for the next mandatory.

Future Improvements, Mandatory 2 Preparation

Honestly, not very proud of my work. I would, If I could do it all again for the second mand. The database is a bit too simple to implement the complex stored procedures required, or maybe I don't have enough imagination or brain power to think of them. The to do list is to find a better way to do the triggers, add a new table if needed to make use of triggers and stored procedures properly. Generally, update the database to be able to handle the feature requirements. As for login functionality, I guess the student, administration, and the teacher, in this case I would probably add grades, so the teacher could update and insert them and the student could see them, also would allow to check if a course is failed and the average grade for that course.