CEN 203 - DATABASE MANAGEMENT SYSTEMS

PROJECT DOCUMENTATION

FACULTY OF ARCHITECTURE AND ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

GROUP 18

Borian Llukaçaj

Engjëll Abazaj

Ermin Lilaj

Iglis Koçiu

Indrit Ferati

Joel Bitri

Kristi Samara (Leader)

PROJECT OUTLINE AND PLANNING

Project Outline

The aim of our project is to create a database similar to Epoka Interactive System (EIS) by following the guidelines and the requirements of the professor. This project involves database design as well as research from the internet and Epoka's staff who has access to EIS, in order to create a well-rounded database.

Project Description

These are the steps we followed to get to the final result:

- Firstly, we did some research about database design by reading and studying our lectures and university materials.
- Secondly, we studied the structure of EIS, by observing the student portal as well as viewing the lecturer portal, in order to gain a full understanding of how the database would be built.
- Then, we started by writing the logic of the database and performing the normalization process on it.
- After that, we designed the ER Diagram which contained entries such as: faculties, departments, programs, lecturers, courses, students etc.
- Based on the ER Diagram we designed the Relational Schema, which shows the relationship between the entries.
- Then, we wrote the code in SQL for creating the structure (tables) of the database and their relationship.
- We then inserted values inside of our database and tested its performance.
- We did some managerial queries on the database.
- In the end, we wrote the documentation for the whole process.

Core Idea

Our goal was to build a database that would resemble the one used for EIS, which holds information about the students and lecturers of Epoka University, as well as to propose potential improvements on the speed and normalization of this database. Our final product manages to describe all the properties of students, lecturers and programs of the university and can actually be implemented in a real-world scenario. This makes it ideal to store and retrieve data efficiently.

RESEARCH AND IDEA CONSOLIDATION

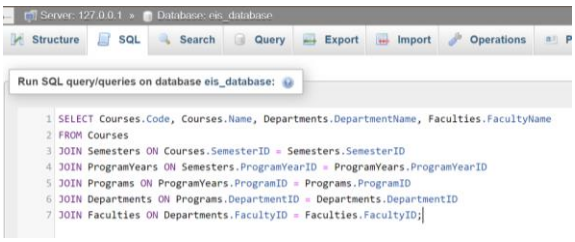
Core Concept of Improvement

The basic concept of improvement involves performing the normalization process. This helps us avoid repetition of data and presents the data in a more organized way. By bringing the tables in the first normal form, we get rid of duplicate entries, the second form allows us to remove the partial key dependencies and the third to remove the transitive key dependencies, thus displaying everything clearer. Since we are dealing with a large database model, after the normalization is finished, we end up with fewer data, better time performance, increased speed and less storage used, which are all positive results.

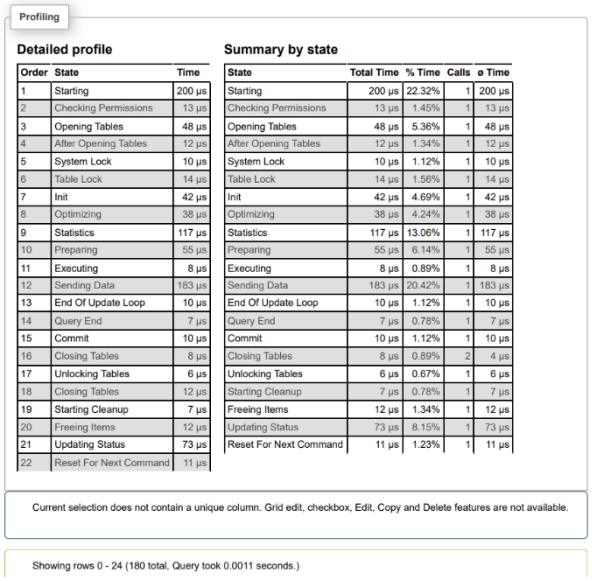
Specifics of Improvement

Our normalization process ensures that no data is duplicated. We improved and accelerated our database by connecting many-to-many and one-to-many relationships and making the appropriate changes. All of this improves performance and facilitates insertion of data and generating queries considerably more quickly and easily. Enhancing the ease of connecting and locating the data improves speed and efficiency. Additionally, we need to consider the number of columns we wish to select, the amount of data, and the number of requirements we are entering. Removing select * and using select with specific columns we wish to pick, or substituting while for having, are two examples of quicker queries.

Concrete Measurement of Query Speed

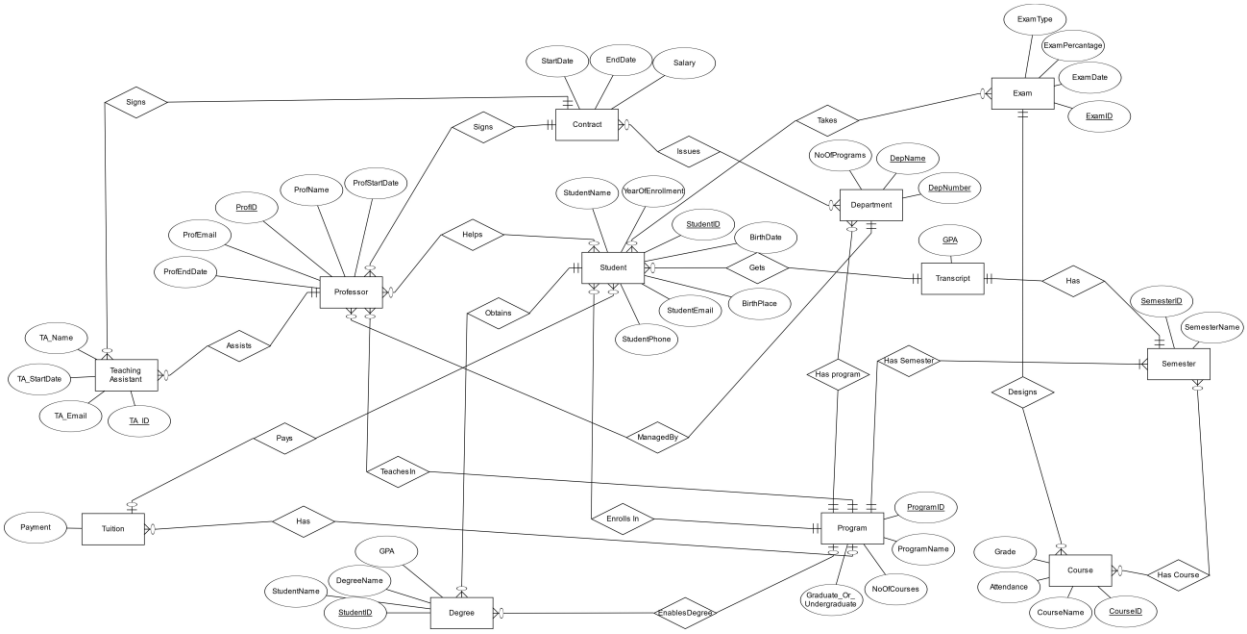


As we can see the execution of the query is quite fast. It takes 0.0011 seconds which shows that the database is fast and efficient.



PROJECT EXECUTION

ER Diagram



ER Diagram Rationales

Our ER diagram starts with the entity of Students and its attributes. Then we created the Department, Program, Course, Exam, Degree, Tuition, Teaching Assistant, Contract, and Professor entities, along with their corresponding attributes.

The Student entity has a many-to-one relationship with the Tuition, where many students may have to pay one and only one tuition, and the tuition may be paid by many students. A one-to-many relationship with the Degree, where each student must obtain one and only one degree, and every degree may be obtained by many students. A many-to-many relationship with Exams, where many students may take many exams and many exams may be taken by many students. A many-to-many relationship with the Professor, where many students may be helped by many professors and many professors may help many students. A many-to-one relationship with Program, where each student must enroll in one and only one program, and programs may be enrolled by many students. Finally, a many-to-one relationship with Transcript, where each student must get one and only one transcript and each transcript may be taken by many students.

The Professor entity has a one-to-many relationship with the Teaching Assistant, where each professor may be assisted by many teaching assistants but each teaching assistant must assist one and only one professor. A many-to-one relationship with Department, where each professor must be managed by one and only one Department, and each department may have many professors. A many-to-one relationship with Program, where each professor must teach in one and only one program, and each program may be taught by many professors. A many-to-one relationship with

Contract, where each professor must sign one and only one contract and each contract may be signed by many professors.

The Teaching Assistant entity has a many-to-one relationship with Contract, where each teaching assistant must sign one and only one contract and each contract may be signed by many teaching assistants.

The Tuition entity has a many-to-one relationship with Program, where each tuition may have one and only one program, and a program may have many tuitions.

The Degree entity has a many-to-one relationship with Program, where each degree may enable degrees to one and only program and each program may have many degrees.

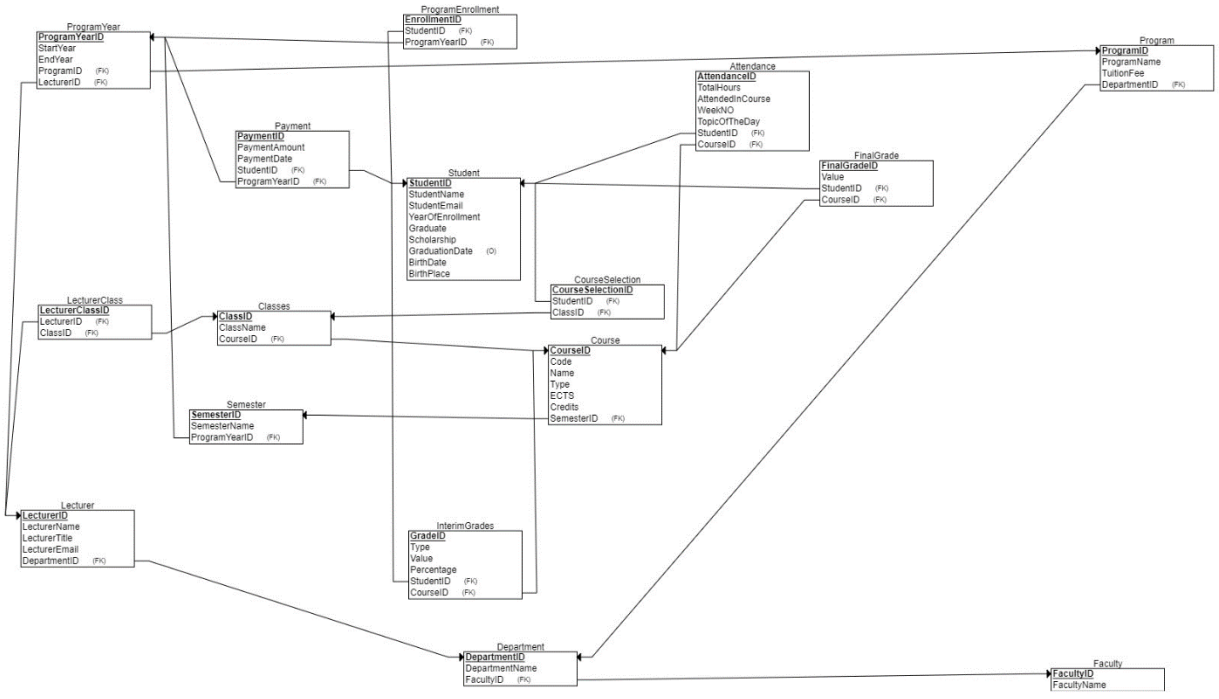
The Program entity has a one-to-many relationship with Department, where each program may have many departments but each department must have one and only one program. A one-to-one relationship with Semester, where each program must have many semesters and each semester must have one and only one program.

The Contract entity has a many-to-many relationship with Department, where each contract may have many departments and each department may have many contracts.

The Course entity has a many-to-many relationship with Semester, where each course may have many semesters and each semester may have many courses. A many-to-one relationship with Exam, where each course must design one and only one exam and each exam may have many courses.

The Semester entity has a one-to-one relationship with Transcript, where each semester must have one and only one transcript and each transcript must have one and only one semester.

RS Schema



RS Schema Rationales

Creating the RS schema after having the ER diagram was a quick and easy step. At the start, every entity and its attributes were converted to a table and then created the relationships one by one. In the case of a one-to-many relationship, the foreign key went to the entity which had many relationships connected to it. In the case of a one-to-one relationship, we did it the way we thought was more logical and easier. In the case of a many-to-many relationship we created another table for that relationship which holds the primary keys of the two tables it connected as primary keys and foreign keys.

Database Dump

CREATE DATA

```sql
CREATE TABLE IF NOT EXISTS Faculties (
    FacultyID INT PRIMARY KEY,
    FacultyName VARCHAR(40) NOT NULL
);
CREATE TABLE IF NOT EXISTS Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(45) NOT NULL,
    FacultyID INT NOT NULL,
    FOREIGN KEY (FacultyID) REFERENCES Faculties(FacultyID)
);
CREATE TABLE IF NOT EXISTS Programs(
    ProgramID INT PRIMARY KEY,
    ProgramName VARCHAR(50) NOT NULL,
    Tuition_Fee INT NOT NULL,
    DepartmentID INT NOT NULL ,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
CREATE TABLE IF NOT EXISTS Lecturers (
    LecturerID INT PRIMARY KEY,
    LecturerName VARCHAR(30) NOT NULL,
    Email VARCHAR(35) NOT NULL,
    DepartmentID INT NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
CREATE TABLE IF NOT EXISTS ProgramYears(
    ProgramYearID INT PRIMARY KEY,
    StartYear INT NOT NULL,
    EndYear INT NOT NULL,
    ProgramID INT NOT NULL,
    FOREIGN KEY (ProgramID) REFERENCES Programs(ProgramID),
    AdvisorID INT NOT NULL,
    FOREIGN KEY (AdvisorID) REFERENCES Lecturers(LecturerID)
);
```

INSERT DATA

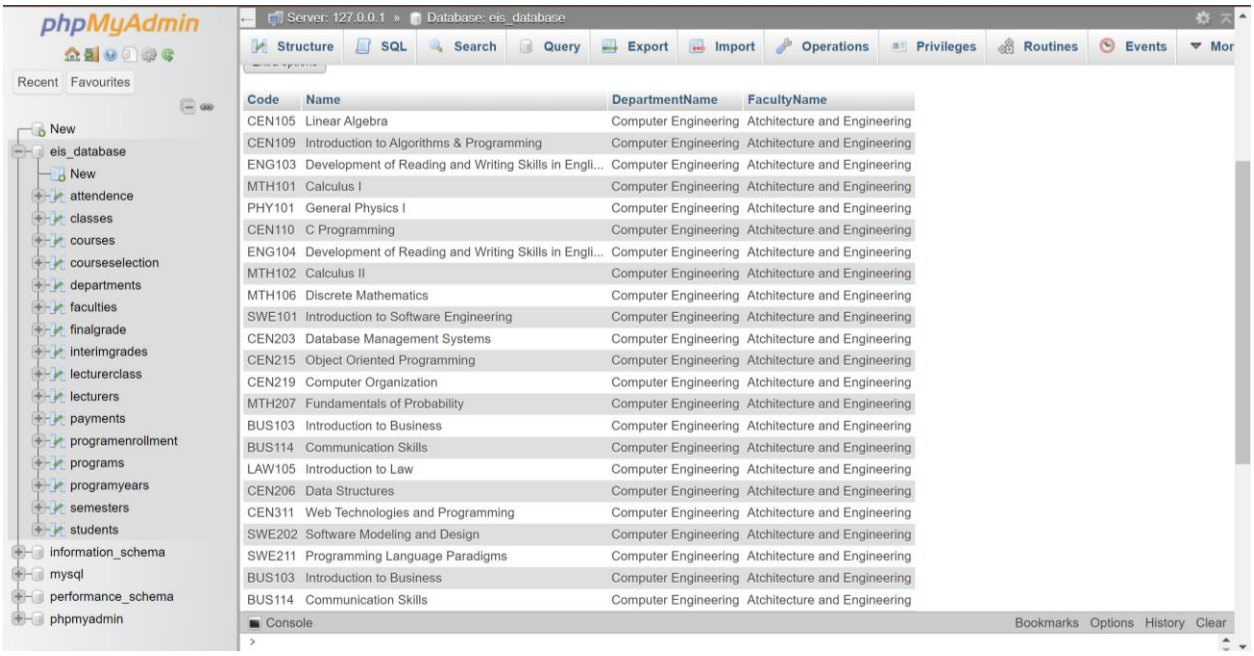
```
188  INSERT INTO Faculties(FacultyID,FacultyName)
189  VALUES (1,'Atchitecture and Engineering'),
190         (2,'Economics and Administrative SC'),
191         (3,'Law and Social SC');
192
193  INSERT INTO Departments(DepartmentID, DepartmentName, FacultyID)
194  VALUES (1,'Civil Engineering',1),
195     (2,'Architecture',1),
196     (3,'Computer Engineering',1),
197     (4,'Economics',2),
198     (5,'Banking and Finance',2),
199     (6,'Business Administration',2),
200     (7,'Political Science and International Relations',3),
201     (8,'Law',3),
202     (9,'Center for European Studies',3);
203
204
205  INSERT INTO Programs(ProgramID, ProgramName, Tuition_Fee, DepartmentID)
206  VALUES (1,'Civil Engineering',3500,1),
207     (2,'Architecture',3800,2),
208     (3,'Computer Engineering',3500,3),
209     (4,'Electronics and Digital Communication Engineering',3500,3),
210     (5,'Software Engineering',4000,3),
211     (6,'Economics',2500,4),
212     (7,'Banking and Finance',2500,5),
213     (8,'Banking and Finance(Albanian)',2500,5),
214     (9,'Business Administration',2500,6),
215     (10,'Business Informatics',3000,6),
216     (11,'International Marketing and Logistics Management',2500,6),
217     (12,'Political Science and International Relations',2500,7),
218     (13,'Law',2800,8);
```

*Please note that the rest of the code will be submitted together with the pdf document.

MANAGERIAL QUERIES

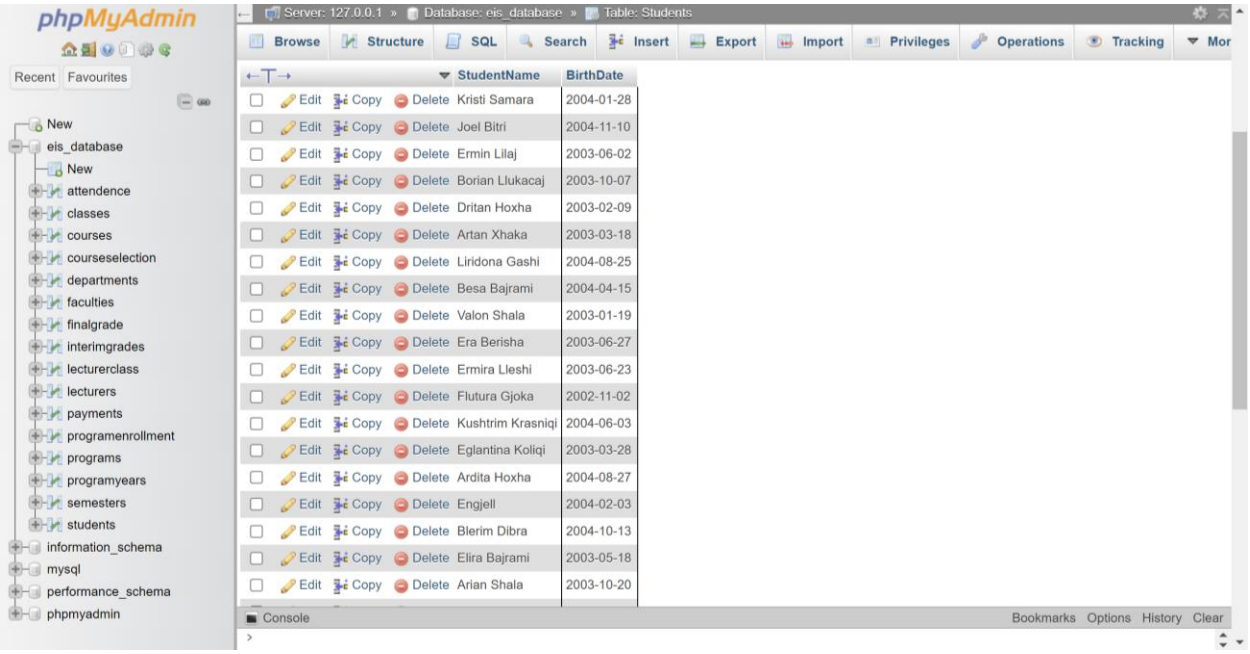
1. Displaying information about the courses

```sql
SELECT Courses.Code, Courses.Name, Departments.DepartmentName, Faculties.FacultyName
FROM Courses
JOIN Semesters ON Courses.SemesterID = Semesters.SemesterID
JOIN ProgramYears ON Semesters.ProgramYearID = ProgramYears.ProgramYearID
JOIN Programs ON ProgramYears.ProgramID = Programs.ProgramID
JOIN Departments ON Programs.DepartmentID = Departments.DepartmentID
JOIN Faculties ON Departments.FacultyID = Faculties.FacultyID;
```



2. Showing the students born after the given year

```sql
SELECT StudentName, BirthDate
FROM Students
WHERE BirthDate > '2002-10-29';
```



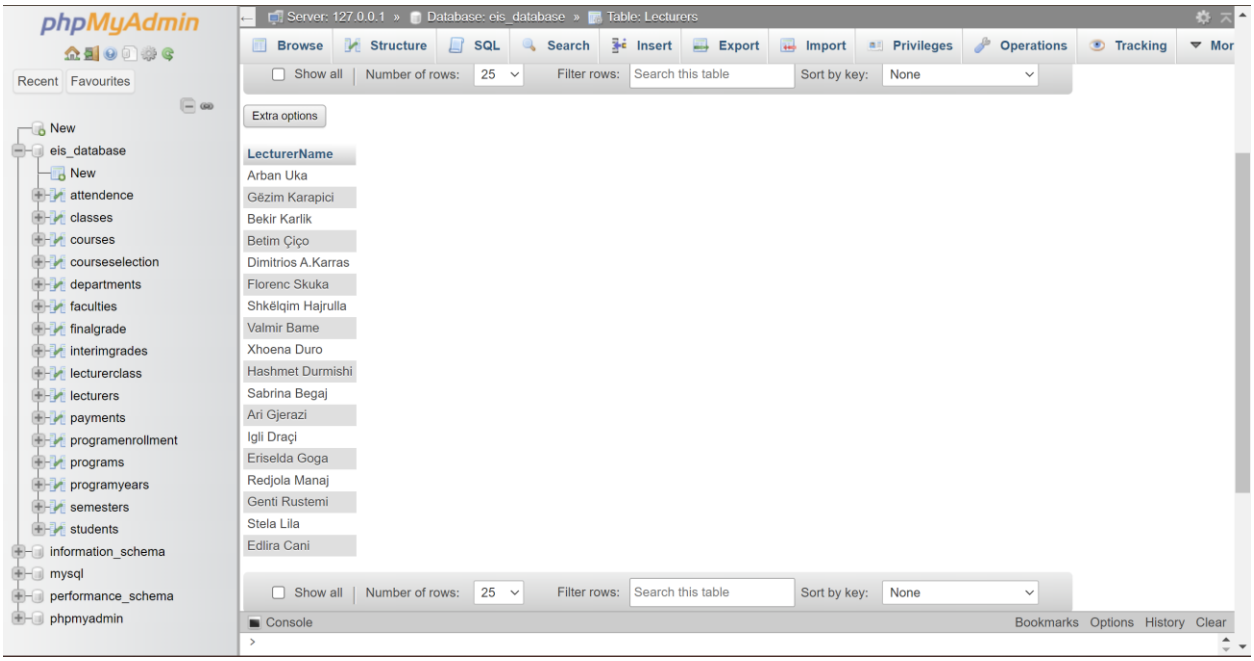
3. Counting the number of students enrolled in each semester

```sql
SELECT Semesters.SemesterName, COUNT(ProgramEnrollment.StudentID) AS StudentCount
FROM Semesters
LEFT JOIN ProgramEnrollment ON Semesters.ProgramYearID = ProgramEnrollment.ProgramYearID
GROUP BY Semesters.SemesterName;
```

SemesterName Fall or spring	StudentCount
Fall	21
Spring	21

4. Displaying lecturers who teach in the given department

```sql
SELECT Lecturers.LecturerName
FROM Lecturers
JOIN Departments ON Lecturers.DepartmentID = Departments.DepartmentID
WHERE Departments.DepartmentName = 'Computer Engineering';
```



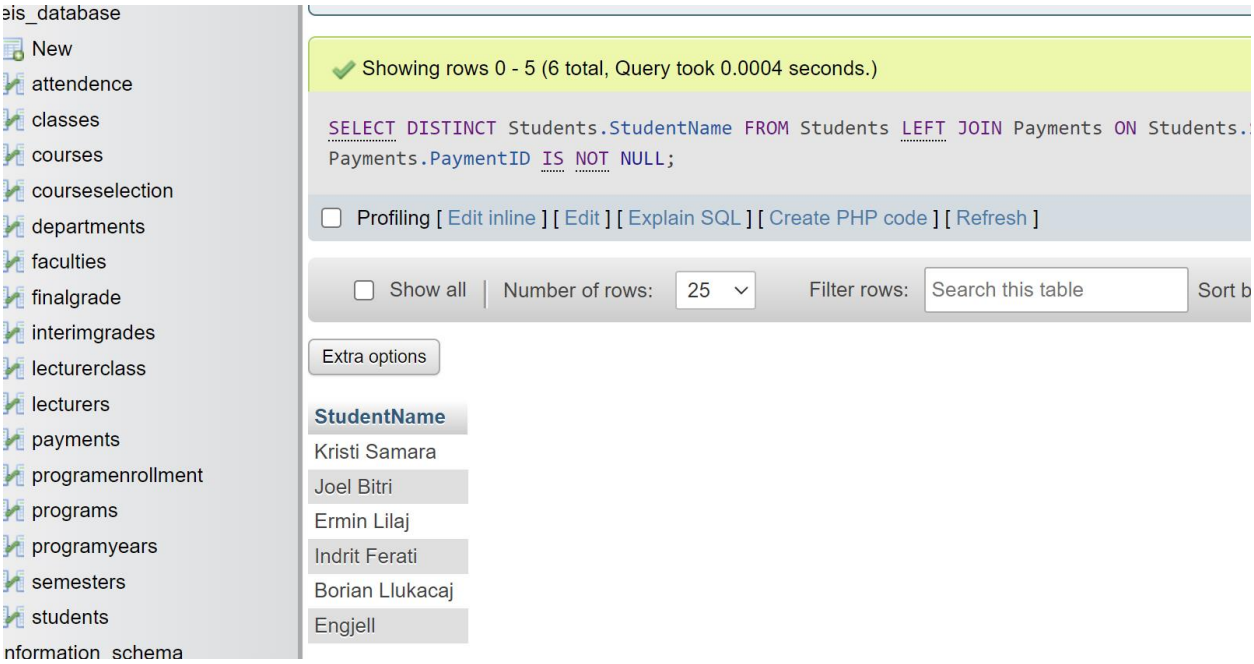
5. Showing the most expensive (max) tuition fee

```sql
SELECT MAX(Tuition_Fee) AS MaxTuitionFee
FROM Programs;
```

MaxTuitionFee
4000

6. Showing students who have paid their tuition fees

```sql
SELECT DISTINCT Students.StudentName
FROM Students
LEFT JOIN Payments ON Students.StudentID = Payments.StudentID
WHERE Payments.PaymentID IS  NOT NULL;
```



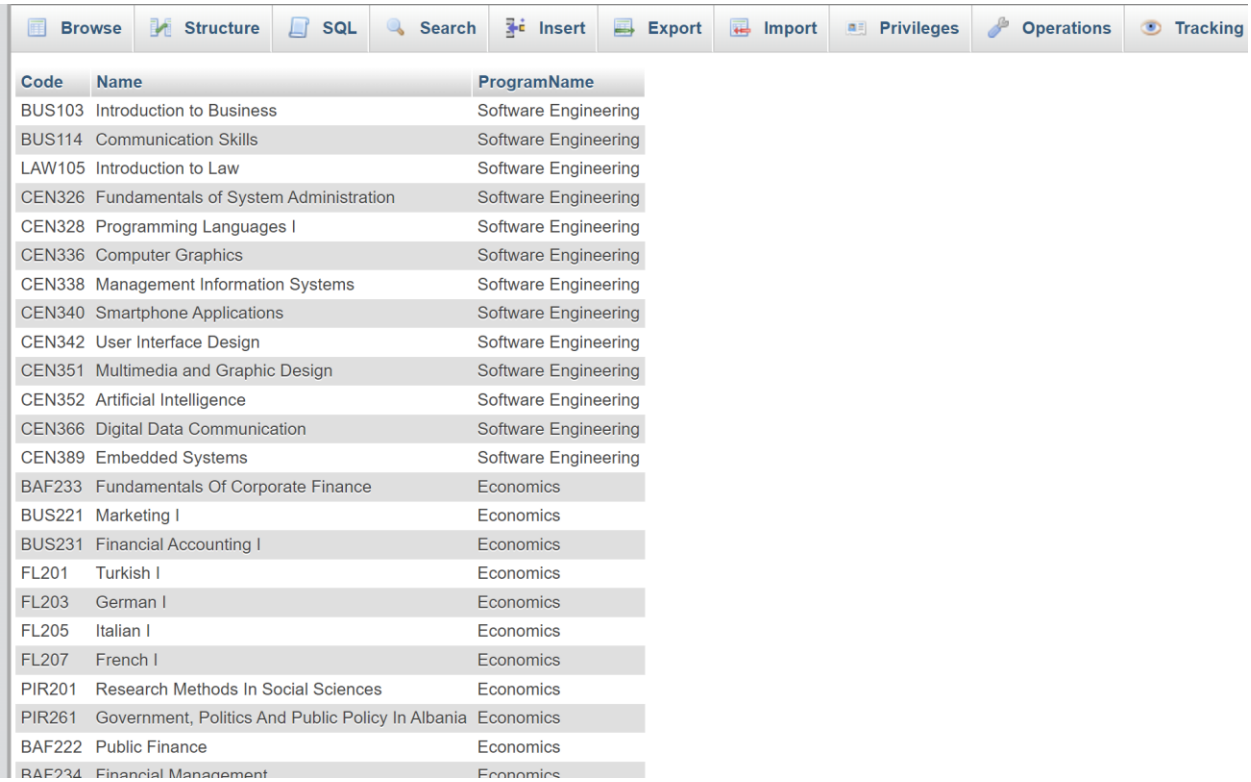
7. Displaying each program and which department it belong to

```sql
SELECT Programs.ProgramName, Departments.DepartmentName
FROM Programs
JOIN Departments ON Programs.DepartmentID = Departments.DepartmentID;
```

ProgramName	DepartmentName
Civil Engineering	Civil Engineering
Architecture	Architecture
Computer Engineering	Computer Engineering
Electronics and Digital Communication Engineering	Computer Engineering
Software Engineering	Computer Engineering
Economics	Economics
Banking and Finance	Banking and Finance
Banking and Finance(Albanian)	Banking and Finance
Business Administration	Business Administration
Business Informatics	Business Administration
International Marketing and Logistics Management	Business Administration
Political Science and International Relations	Political Science and International Relations
Law	Law

8. Showing all elective courses

```sql
SELECT DISTINCT Courses.Code, Courses.Name, Programs.ProgramName
FROM Courses
JOIN Semesters ON Courses.SemesterID = Semesters.SemesterID
JOIN ProgramYears ON Semesters.ProgramYearID = ProgramYears.ProgramYearID
JOIN Programs ON ProgramYears.ProgramID = Programs.ProgramID
WHERE Courses.Type = 'Elective';
```



Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking

Code	Name	ProgramName
BUS103	Introduction to Business	Software Engineering
BUS114	Communication Skills	Software Engineering
LAW105	Introduction to Law	Software Engineering
CEN326	Fundamentals of System Administration	Software Engineering
CEN328	Programming Languages I	Software Engineering
CEN336	Computer Graphics	Software Engineering
CEN338	Management Information Systems	Software Engineering
CEN340	Smartphone Applications	Software Engineering
CEN342	User Interface Design	Software Engineering
CEN351	Multimedia and Graphic Design	Software Engineering
CEN352	Artificial Intelligence	Software Engineering
CEN366	Digital Data Communication	Software Engineering
CEN389	Embedded Systems	Software Engineering
BAF233	Fundamentals Of Corporate Finance	Economics
BUS221	Marketing I	Economics
BUS231	Financial Accounting I	Economics
FL201	Turkish I	Economics
FL203	German I	Economics
FL205	Italian I	Economics
FL207	French I	Economics
PIR201	Research Methods In Social Sciences	Economics
PIR261	Government, Politics And Public Policy In Albania	Economics
BAF222	Public Finance	Economics
BAF234	Financial Management	Economics

9. Displaying all the courses taught by a given lecturer

```sql
SELECT Lecturers.LecturerName, Courses.Code, Courses.Name
FROM LecturerClass
JOIN Lecturers ON LecturerClass.LecturerID = Lecturers.LecturerID
JOIN Classes ON LecturerClass.ClassID = Classes.ClassID
JOIN Courses ON Classes.CourseID = Courses.CourseID
WHERE Lecturers.LecturerName = 'Sabrina Begaj';
```

LecturerName	Code	Name
Sabrina Begaj	CEN215	Object Oriented Programming
Sabrina Begaj	CEN219	Computer Organization
Sabrina Begaj	CEN219	Computer Organization

10. Showing how many students are in each academic year

```sql
SELECT ProgramYears.StartYear, COUNT(ProgramEnrollment.StudentID) AS StudentCount
FROM ProgramYears
LEFT JOIN ProgramEnrollment ON ProgramYears.ProgramYearID = ProgramEnrollment.ProgramYearID
GROUP BY ProgramYears.StartYear;
```

StartYear	StudentCount
2021	3
2022	14
2023	4

11. Finding the department with the biggest number of students

```sql
SELECT d.DepartmentName, COUNT(pe.StudentID) AS EnrolledStudents
FROM Departments d
JOIN Programs p ON d.DepartmentID = p.DepartmentID
JOIN ProgramYears py ON p.ProgramID = py.ProgramID
JOIN ProgramEnrollment pe ON py.ProgramYearID = pe.ProgramYearID
GROUP BY d.DepartmentName
ORDER BY EnrolledStudents DESC
LIMIT 1;
```

DepartmentName	EnrolledStudents
Computer Engineering	10

12. Top 5 lecturers with the biggest number of students in their classes

```sql
SELECT l.LecturerName, COUNT(cs.StudentID) AS EnrolledStudents
FROM Lecturers l
JOIN LecturerClass lc ON l.LecturerID = lc.LecturerID
JOIN Classes c ON lc.ClassID = c.ClassID
JOIN CourseSelection cs ON c.ClassID = cs.ClassID
GROUP BY l.LecturerName
ORDER BY EnrolledStudents DESC
LIMIT 5;
```

LecturerName	EnrolledStudents ▾ 1
Sabrina Begaj	9
Ari Gjerazi	6
Eriselda Goga	6
Igli Draçi	6
Redjola Manaj	6

13. Lecturer who has taught most classes

```sql
SELECT l.LecturerName, COUNT(lc.ClassID) AS TaughtClassesCount
FROM Lecturers l
JOIN LecturerClass lc ON l.LecturerID = lc.LecturerID
GROUP BY l.LecturerName
ORDER BY TaughtClassesCount DESC
LIMIT 1;
```

LecturerName	TaughtClassesCount
Sabrina Begaj	3

14. Displaying students that have a scholarship

```sql
SELECT StudentName, Scolarship
FROM Students
WHERE Scolarship > 0
ORDER BY Scolarship DESC;
```

	StudentName	Scolarship 1
Edit Copy Delete	Eglantina Koliqi	100
Edit Copy Delete	Edmond Xhaka	100
Edit Copy Delete	Shkelqim Dibra	100
Edit Copy Delete	Engjell	75
Edit Copy Delete	Lum Bajrami	75
Edit Copy Delete	Ermira Lleshi	75
Edit Copy Delete	Valon Shala	50
Edit Copy Delete	Arian Shala	50
Edit Copy Delete	Ermin Lilaj	50
Edit Copy Delete	Kristi Samara	10

15. Total credits earned by each student

```sql
SELECT s.StudentID, s.StudentName, SUM(c.Credits) AS TotalCreditsEarned
FROM Students s
JOIN CourseSelection cs ON s.StudentID = cs.StudentID
JOIN Classes cl ON cs.ClassID = cl.ClassID
JOIN Courses c ON cl.CourseID = c.CourseID
GROUP BY s.StudentID, s.StudentName;
```

StudentID	StudentName	TotalCreditsEarned
1	Kristi Samara	17
2	Joel Bitri	17
3	Ermin Lilaj	17
4	Indrit Ferati	17
5	Borian Llukacaj	17
12	Valon Shala	4
21	Engjell	17

16. Students who have paid more than the average payment

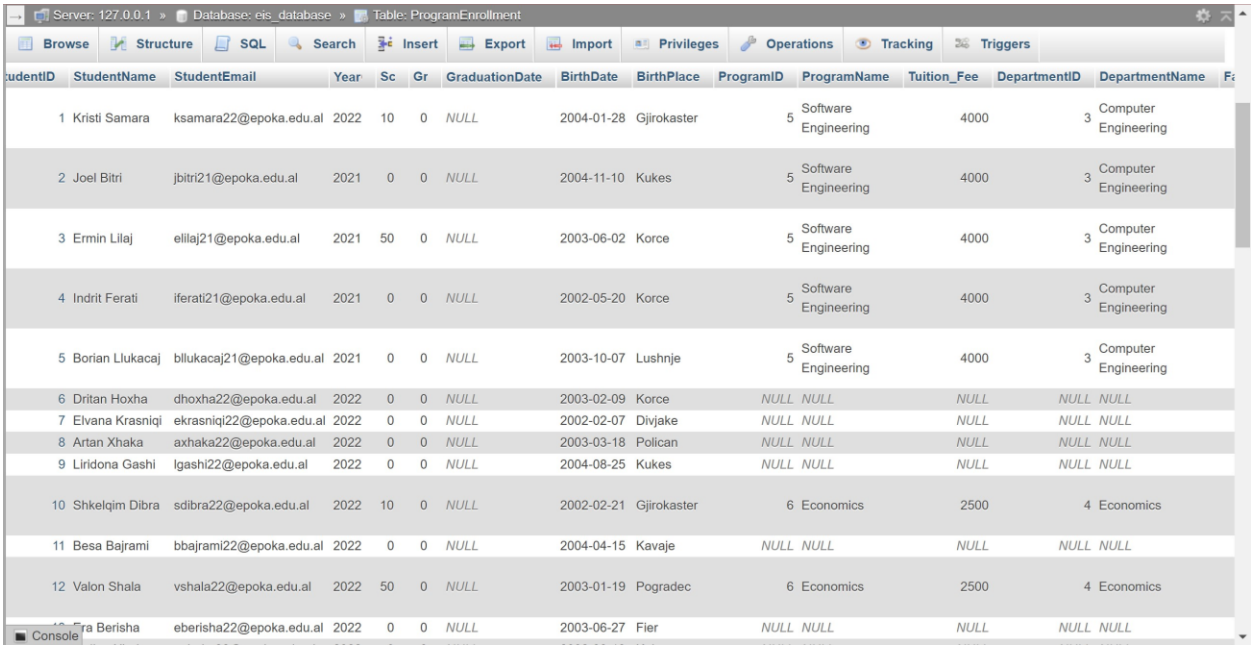
```sql
SELECT DISTINCT s.StudentName, p.PaymentAmount, py.ProgramYearID
FROM Students s
JOIN Payments p ON s.StudentID = p.StudentID
JOIN ProgramEnrollment pe ON s.StudentID = pe.StudentID
JOIN ProgramYears py ON pe.ProgramYearID = py.ProgramYearID
WHERE p.PaymentAmount > (
    SELECT AVG(p2.PaymentAmount)
    FROM Payments p2
    WHERE p2.ProgramYearID = py.ProgramYearID
);
```

StudentName	PaymentAmount	ProgramYearID
Kristi Samara	1800	20
Joel Bitri	2000	20
Indrit Ferati	2000	20

17. Retrieving a lot of information from the database

```sql
SELECT
    s.StudentID,
    s.StudentName,
    s.StudentEmail,
    s.YearOfEnrollment,
    s.Scolarship,
    s.Graduate,
    s.GraduationDate,
    s.BirthDate,
    s.BirthPlace,
    pr.ProgramID,
    pr.ProgramName,
    pr.Tuition_Fee,
    d.DepartmentID,
    d.DepartmentName,
```

```sql
    f.FacultyID,
    f.FacultyName
FROM
    Students s
LEFT JOIN
    ProgramEnrollment pe ON s.StudentID = pe.StudentID
LEFT JOIN
    ProgramYears py ON pe.ProgramYearID = py.ProgramYearID
LEFT JOIN
    Programs pr ON py.ProgramID = pr.ProgramID
LEFT JOIN
    Departments d ON pr.DepartmentID = d.DepartmentID
LEFT JOIN
    Faculties f ON d.FacultyID = f.FacultyID;
```



udentID	StudentName	StudentEmail	Year	Sc	Gr	GraduationDate	BirthDate	BirthPlace	ProgramID	ProgramName	Tuition_Fee	DepartmentID	DepartmentName	Fa
1	Kristi Samara	ksamara22@epoka.edu.al	2022	10	0	NULL	2004-01-28	Gjirokaster	5	Software Engineering	4000	3	Computer Engineering	
2	Joel Bitri	jbitri21@epoka.edu.al	2021	0	0	NULL	2004-11-10	Kukes	5	Software Engineering	4000	3	Computer Engineering	
3	Ermin Lilaj	elilaj21@epoka.edu.al	2021	50	0	NULL	2003-06-02	Korce	5	Software Engineering	4000	3	Computer Engineering	
4	Indrit Ferati	iferati21@epoka.edu.al	2021	0	0	NULL	2002-05-20	Korce	5	Software Engineering	4000	3	Computer Engineering	
5	Borian Llukacaj	bllukacaj21@epoka.edu.al	2021	0	0	NULL	2003-10-07	Lushnje	5	Software Engineering	4000	3	Computer Engineering	
6	Dritan Hoxha	dhoxha22@epoka.edu.al	2022	0	0	NULL	2003-02-09	Korce	NULL	NULL	NULL	NULL	NULL	
7	Elvana Krasniqi	ekrasniqi22@epoka.edu.al	2022	0	0	NULL	2002-02-07	Divjake	NULL	NULL	NULL	NULL	NULL	
8	Artan Xhaka	axhaka22@epoka.edu.al	2022	0	0	NULL	2003-03-18	Polican	NULL	NULL	NULL	NULL	NULL	
9	Liridona Gashi	lgashi22@epoka.edu.al	2022	0	0	NULL	2004-08-25	Kukes	NULL	NULL	NULL	NULL	NULL	
10	Shkelqim Dibra	sdibra22@epoka.edu.al	2022	10	0	NULL	2002-02-21	Gjirokaster	6	Economics	2500	4	Economics	
11	Besa Bajrami	bbajrami22@epoka.edu.al	2022	0	0	NULL	2004-04-15	Kavaje	NULL	NULL	NULL	NULL	NULL	
12	Valon Shala	vshala22@epoka.edu.al	2022	50	0	NULL	2003-01-19	Pogradec	6	Economics	2500	4	Economics	
	Era Berisha	eberisha22@epoka.edu.al	2022	0	0	NULL	2003-06-27	Fier	NULL	NULL	NULL	NULL	NULL	

18. Displaying the grades of a given student

```sql
SELECT ig.GradeID, ig.Type, ig.Percentage, ig.Val, c.Name AS CourseName
FROM InterimGrades ig
JOIN Courses c ON ig.CourseID = c.CourseID
JOIN Students s ON ig.StudentID = s.StudentID
WHERE s.StudentID = 21;
```

GradeID	Type	Percentage	Val	CourseName
101	final	45	93	Database Management Systems
102	midterm	25	65	Database Management Systems
103	assigments	10	67	Database Management Systems
104	quiz	20	66	Database Management Systems
105	final	45	32	Object Oriented Programming
106	midterm	25	0	Object Oriented Programming
107	assigments	10	40	Object Oriented Programming
108	quiz	20	87	Object Oriented Programming
109	final	45	16	Computer Organization
110	midterm	25	98	Computer Organization
111	assigments	10	7	Computer Organization
112	quiz	20	16	Computer Organization
113	final	45	77	Fundamentals of Probability
114	midterm	25	52	Fundamentals of Probability
115	assigments	10	73	Fundamentals of Probability
116	quiz	20	6	Fundamentals of Probability
117	final	45	63	Introduction to Business
118	midterm	25	50	Introduction to Business
119	assigments	10	73	Introduction to Business
120	quiz	20	85	Introduction to Business

19. Calculating average attendance percentage for each course

```
SELECT C.CourseID, C.Name AS CourseName, AVG(A.attended * 100.0 / A.total) AS
AvgAttendancePercentage
FROM Courses C
LEFT JOIN Attendence A ON C.CourseID = A.CourseID
GROUP BY C.CourseID, C.Name
HAVING AVG(A.attended * 100.0 / A.total) < 70;
```

CourseID	CourseName	AvgAttendancePercentage
11	Database Management Systems	52.564102564
12	Object Oriented Programming	53.418803525
13	Computer Organization	50.000000000
14	Fundamentals of Probability	50.641025641
15	Introduction to Business	51.282051153

20. Showing students who are enrolled in a program and lecturers who are teaching a course

```
SELECT StudentName AS Name, StudentEmail AS Email, 'Enrolled in Program' AS Details
FROM Students
UNION
SELECT LecturerName AS Name, Email AS Email, 'Teaching Course' AS Details
FROM Lecturers;
```

Loreta Gashi	lgashi22@epoka.edu.al	Enrolled in Program
Endrit Dibra	edibra22@epoka.edu.al	Enrolled in Program
Rina Shala	rshala22@epoka.edu.al	Enrolled in Program
Enea Berisha	eberisha22@epoka.edu.al	Enrolled in Program
Ardit Xhaka	axhaka22@epoka.edu.al	Enrolled in Program
Besa Lleshi	blleshi22@epoka.edu.al	Enrolled in Program
Diona Kuka	dkuka22@epoka.edu.al	Enrolled in Program
Fation Gjoka	fgjoka22@epoka.edu.al	Enrolled in Program
Klea Krasniqi	kkrasniqi22@epoka.edu.al	Enrolled in Program
Elda Koliqi	ekoliqi22@epoka.edu.al	Enrolled in Program
Rron Hoxha	rhoxha22@epoka.edu.al	Enrolled in Program
Nertil Mera	nmera@epoka.edu.al	Teaching Course
Chrysanthi Balomenou	cbalomenou@epoka.edu.al	Teaching Course
Fatbardha Morina	fmorina@epoka.edu.al	Teaching Course
Armanda Tola	atola@epoka.edu.al	Teaching Course
Albina Hysaj	ahysaj@epoka.edu.al	Teaching Course
Elvira Meti	emeti@epoka.edu.al	Teaching Course
Dafina Muda	dshehi@epoka.edu.al	Teaching Course
Esmir Demaj	edemaj@epoka.edu.al	Teaching Course
Osman Aras	oaras@epoka.edu.al	Teaching Course