UNIVERSITY EDUCATION DATABASE SYSTEM

1)TOPIC AND MEMBER-TASK RESPONSIBILITIES

- **Topic** : Our project topic is Education Database System .

  →There are faculties and departments.
  →A faculty can be associated with more than one department.
  →Each department has instructors and students.
  →Students enroll the courses.
  →Courses make exams.
  →Sections has date.
  →A course can be associated with more than one section.
  →An instructor can work for only one department.
  →Instructors give courses.

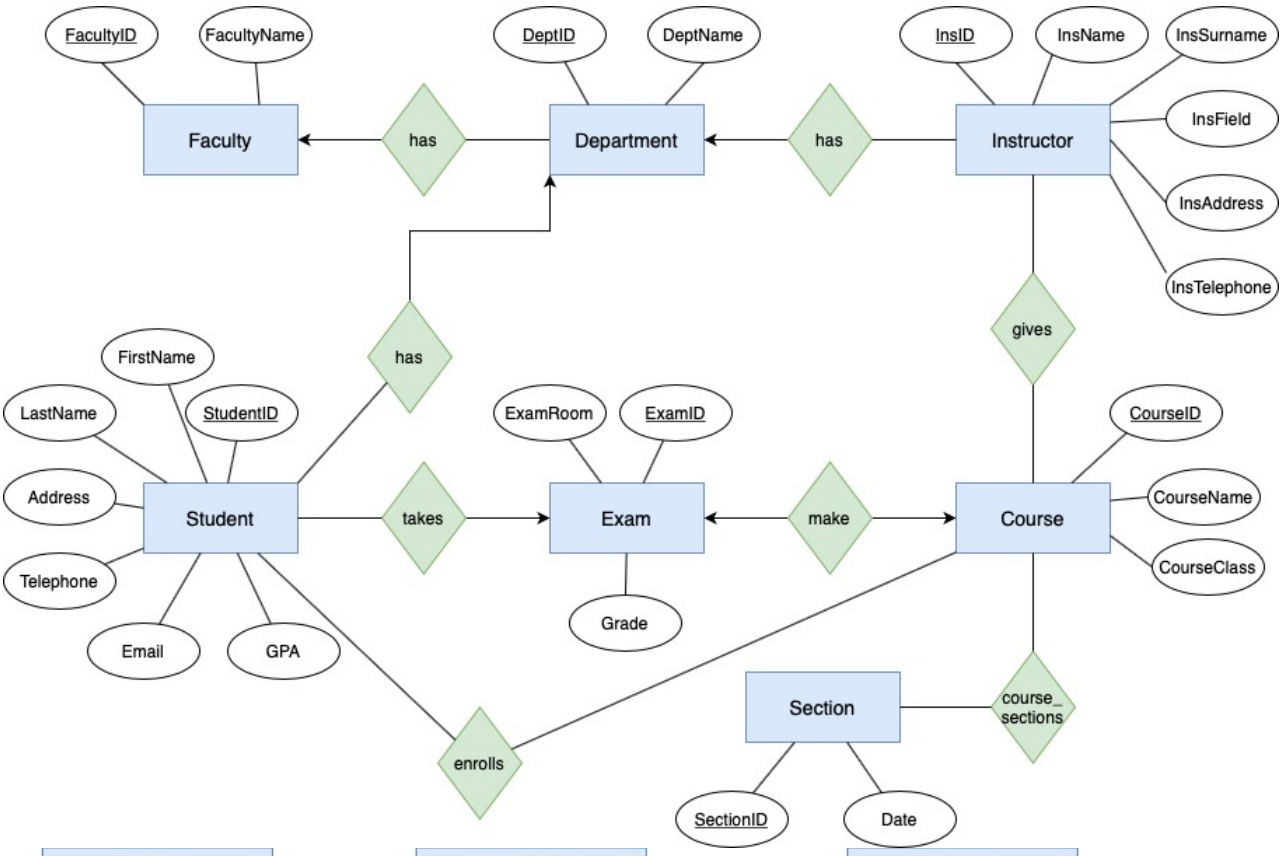
- **Member-Task Responsibilities** : We divided our project into three parts. Basically,preparing diagram and schema , creating and filling a table , writing different SQL queries.

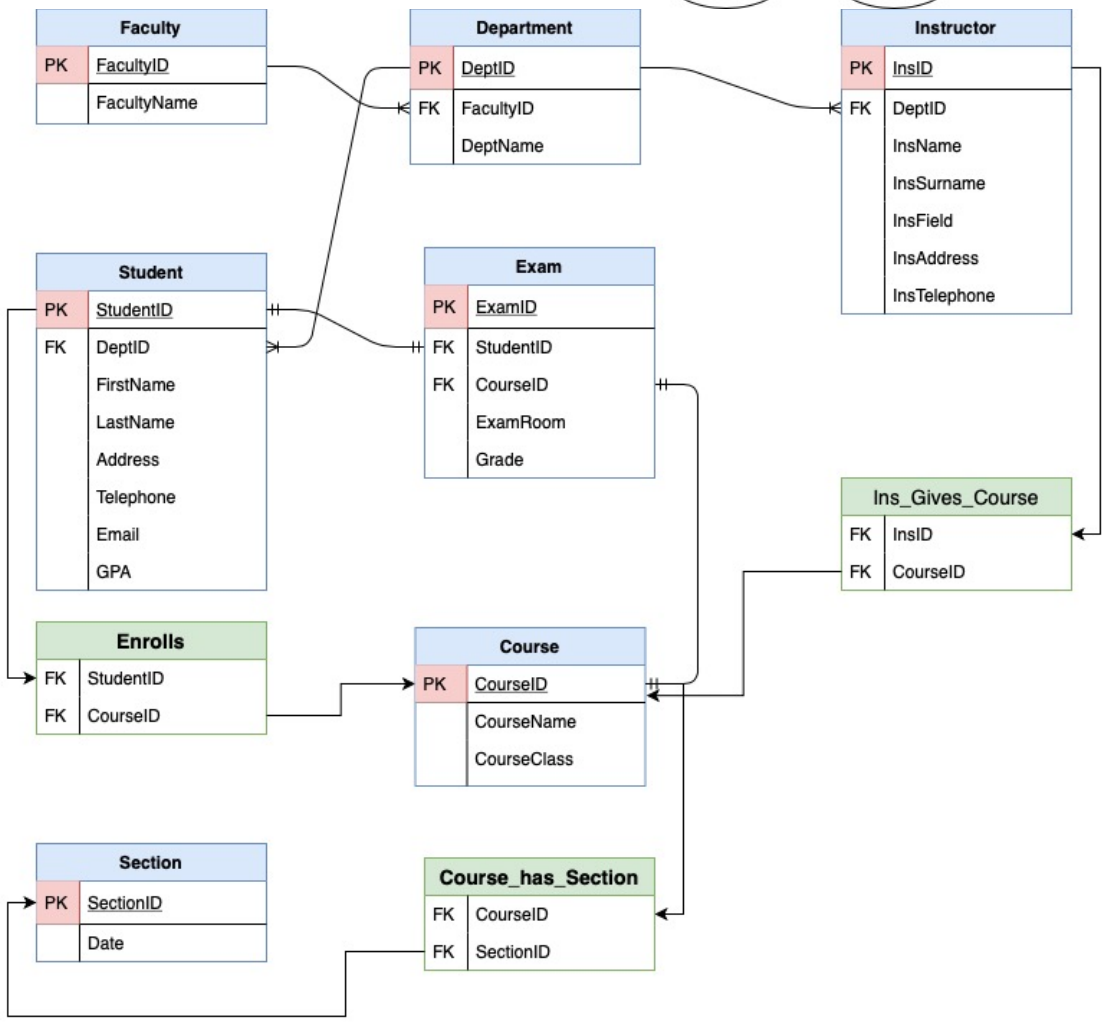
  →Yiğithan Yurtseven(1803715) : He designed E-R diagram based on our project topic and drew a database schema diagram to our E-R design.

  →Aslı Balın(1803626) : She wrote CREATE TABLE and INSERT INTO statements for each table.

  →Hande Aksu(1804828) : She wrote 10 different SQL queries for the database we created.

2) E-R DIAGRAM AND DATABASE SCHEMA DIAGRAM





3)CREATE TABLE AND INSERT INTO STATEMENTS

```
CREATE TABLE Faculty(
FacultyID number(5)  primary key,
FacultyName varchar2(50) ) ;

CREATE TABLE Department(
DeptID number(5) primary key,
DeptName varchar2(50),
FacultyID number(5) references Faculty(FacultyID) );


CREATE TABLE Student(
StudentID number(5) primary key,
DeptID number(5)  references Department(DeptID),
FirstName varchar2(30) ,
LastName varchar2(30) ,
Address varchar2(50) ,
Telephone varchar(12) ,
Email varchar(40) ,
GPA float );

CREATE TABLE Instructor(
InsID number(5)   primary key,
DeptID number(5)  references Department(DeptID),
InsName varchar2(30),
InsSurname varchar2(30),
InsField varchar2(30),
InsAddress varchar2(50),
InsTelephone varchar(10) ) ;

CREATE TABLE Course(
CourseID number(5) primary key,
CourseName varchar2(30),
CourseClass varchar2(10) );

CREATE TABLE Exam(
ExamID number(2)  primary key,
StudentID number(5)  references Student(StudentID),
CourseID number(5)  references Course(CourseID),
ExamRoom varchar2(4),
Grade varchar(3) );

CREATE TABLE Section(
SectionID number(1) primary key,
sectionDate date ) ;
```

```
CREATE TABLE Enrolls(
StudentID number(5),
CourseID number(5),
constraint enr_stID_fk foreign key(StudentID) references Student(StudentID)  );

CREATE TABLE Ins_Gives_Course(
InsID number(5) references Instructor(InsID),
CourseID number(5) references Course(CourseID));


CREATE TABLE Course_has_Section(
CourseID number(5) references Course(CourseID),
SectionID number(5) references Section(SectionID) );


INSERT INTO Faculty VALUES (12345 ,    ' Faculty of Engineering and Natural Sciences'    );
INSERT INTO Faculty VALUES (23456 ,    'Faculty of Health Sciences'    );
INSERT INTO Faculty VALUES (34567 ,    'Faculty of Architecture and Design'  );

INSERT INTO Department VALUES(10000    , 'Software Engineering' , 12345  );
INSERT INTO Department VALUES(20000    , 'Nutrition and Dietetics' ,23456     );
INSERT INTO Department VALUES(30000    ,'Architecture'   ,   34567   );

INSERT INTO Instructor VALUES(00001  , 10000 ,   'Ariana'  ,   'Grande'   , 'Artificial intelligence', 'Florida'
,  '4536778890' ) ;
INSERT INTO Instructor VALUES(00002 , 20000,    'Lady'  ,   'Gaga'   , 'Nutrition science'  ,   'New York' ,
'2567447898' );
INSERT INTO Instructor VALUES(00003,30000 ,   'Taylor'  , 'Swift'  , 'Architectural design' ,
'Pennsylvania'   ,   '5564091422'   );


INSERT INTO Student VALUES(01000 , 10000 ,  'Aslı'  ,   'Balın'  , 'Istanbul'  , '5995664327' ,
'aslibalin@hotmail.com'   ,   3.54   );
INSERT INTO Student VALUES(02000 , 20000 ,  'Hande' , 'Aksu'   , 'Izmir'  , '5522511102' ,
'handeaksu@hotmail.com'  ,  3.99  );
INSERT INTO Student VALUES(03000 ,  30000 ,  'Yiğithan'  ,  'Yurtseven'  ,  'Ordu'  ,  '5986772395'  ,
'yigithanyurtseven@hotmail.com'   ,   3.85  );

INSERT INTO Course VALUES(01010  ,  ' Database Management Systems'  ,  'A102'  );
INSERT INTO Course VALUES(02020  , 'Principles of Nutrition'  ,   'B305'  );
INSERT INTO Course VALUES(03030  ,   ' Digital Media in Architecture '   ,  'D304'   );

INSERT INTO Exam VALUES( 10  ,  01000  ,  01012 , 'D404'   ,   '70'   );
INSERT INTO Exam VALUES(20  ,   02000  ,  02022,   'A203'    ,    '100'  );
INSERT INTO Exam VALUES(30  ,   03000 ,   03030,   'B204'    ,   '80'   );

INSERT INTO Section VALUES(1 ,   TO_DATE ( '10/11/2020 ' , 'DD/MM/YYYY' ) );
INSERT INTO Section VALUES(2 ,   TO_DATE ( '15/11/2020' , 'DD/MM/YYYY' ) );
INSERT INTO Section VALUES(3 ,   TO_DATE ( '20/11/2020' , 'DD/MM/YYYY' ) );
```

4)SQL QUERIES

- Find out how many students are in which department.
  → **AGGREGATE OPERATION**

  SELECT deptid AS Department_ID,deptname as Department_Name,count(studentID) as Number_Of_Students
  FROM student NATURAL JOIN department
  GROUP BY deptid,deptname;

DEPARTMENT_ID	DEPARTMENT_NAME	NUMBER_OF_STUDENTS
30000	Architecture	1
20000	Nutrition and Dietetics	1
10000	Software Engineering	1

- Find the exam average higher than 50 for each course.
  → **AGGREGATE OPERATION**

  SELECT c.coursename AS Course_Name,avg(grade) AS Average_Grade
  FROM exam e, course c
  WHERE e.courseid=c.courseid
  GROUP BY coursename
  HAVING avg(grade) > 50;

COURSE_NAME	AVERAGE_GRADE
Principles of Nutrition	100
Digital Media in Architecture	80
Database Management Systems	70

- Find the phone numbers of the students studying in the software engineering department. → **NESTED QUERIES**

```
SELECT firstname ||' '||lastname AS Student_Name,telephone AS
Telephone_Number
FROM student s,department d
WHERE s.deptid=d.deptid and d.deptid =
(SELECT deptid
FROM department
WHERE deptname='Software Engineering');
```

STUDENT_NAME	TELEPHONE_NUMBER
Aslı Balın	5995664327

- Find the names and fields of instructors working in the nutrition and dietetics department. → **NESTED QUERIES**

```
SELECT insname ||' '|| inssurname AS Instructor_FullName , insfield AS
Instructor_Field,d.deptname AS Department_Name
FROM instructor i LEFT OUTER JOIN department d ON i.deptid = d.deptid
WHERE insid IN
(SELECT insid
FROM instructor
WHERE deptname='Nutrition and Dietetics');
```

INSTRUCTOR_FULLNAME	INSTRUCTOR_FIELD	DEPARTMENT_NAME
Lady Gaga	Nutrition science	Nutrition and Dietetics

- Find students studying architecture and averaging over 3.50.
  →**SET OPERATION**

```
SELECT firstname ||' '||lastname AS Student_Name , gpa
FROM student
WHERE deptid IN(
SELECT deptid
FROM department
WHERE deptname ='Architecture' )
INTERSECT
SELECT firstname ||' '||lastname AS Student_Name ,gpa
FROM student
WHERE gpa>3.50;
```

STUDENT_NAME	GPA
Yiğithan Yurtseven	3.85

- Find students whose average is more than 3.50 but do not study in software engineering. →**SET OPERATION**

```
SELECT firstname ||' '||lastname AS Student_FullName , gpa
FROM student
WHERE gpa>3.50
MINUS
SELECT firstname ||' '||lastname AS Student_FullName , gpa
FROM student
WHERE deptid IN(
SELECT deptid
FROM department
WHERE deptname ='Software Engineering' );
```

STUDENT_FULLNAME	GPA
Hande Aksu	3.99
Yiğithan Yurtseven	3.85

- Find the names and IDS of the students who will take the exam in D404. →**JOIN**

  SELECT s.studentid AS Student_Number ,firstname ||' '||lastname AS Student_FullName , e.examroom
  FROM student s INNER JOIN exam e ON s.studentid = e.studentid
  WHERE examroom='D404';

STUDENT_NUMBER	STUDENT_FULLNAME	EXAMROOM
1000	Aslı Balın	D404

- Find the names and numbers of students taking the database management systems course in descending order. → **JOIN**

  SELECT studentid AS Student_Number ,firstname ||' '||lastname AS Student_FullName
  FROM student NATURAL INNER JOIN enrolls
  WHERE courseID=
  (SELECT courseID
  FROM course
  WHERE coursename='Database Management Systems')
  ORDER BY studentid desc;

STUDENT_NUMBER	STUDENT_FULLNAME
2000	Hande Aksu
1000	Aslı Balın

- Find the students whose grades are lower than the exam average.
  →**JOIN**

  SELECT s.studentid AS Student_Number ,firstname ||' '||lastname AS Student_FullName ,gpa , e.grade
  FROM student s , exam e
  WHERE s.studentid(+)=e.studentid and e.grade < SOME
  (SELECT avg(grade)
  FROM exam);

STUDENT_NUMBER	STUDENT_FULLNAME	GPA	GRADE
1000	Aslı Balın	3.54	70
3000	Yiğithan Yurtseven	3.85	80

- Find the names and GPAs of all students that have greater GPAS than all students . → **SELF JOIN**

  SELECT studentid AS Student_ID ,firstname ||' '||lastname AS Student_FullName ,gpa
  FROM student
  WHERE gpa >=ALL
  (SELECT gpa
  FROM student);

STUDENT_ID	STUDENT_FULLNAME	GPA
2000	Hande Aksu	3.99