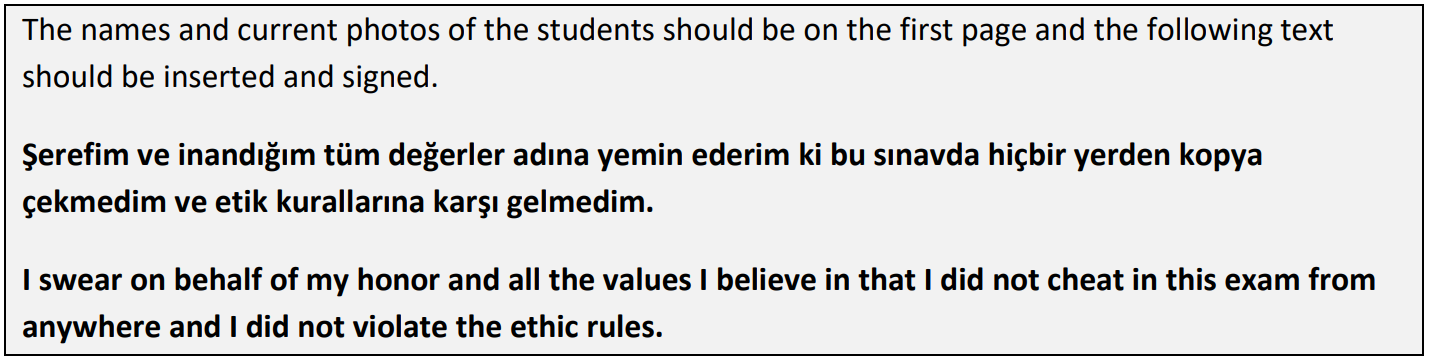
# **BEYKOZ UNIVERSITY DATABASE SYSTEMS**

# **Term Project Phase II**

# Homework Tracking System

**Database Systems - 60612MEEOZ-CME0075-2021-2022-1**

**Lecturer : Selçuk KIRAN**

**

**

*Mert Altuntaş / 1804010005*

**PHASE I**

## **1. Brief Description**

This study includes a new database application that will facilitate the exchange and communication of documents in homework and project weighted courses. With the designed database, they will carry out transactions such as sending files and messaging between teachers and learners through the system. The database infrastructure of the program is MsSql.

Thanks to this project, academicians will spend less time in the homework and project evaluation process and will gain time for their other academic or other works.

### **1.1 Scope Definition**

In the teaching process, the instructors want to control the learning status of the students for application-oriented, graphic, coding or other courses. In order to fulfill this function, it assigns students the responsibility of homework and projects to be delivered between certain dates.

The learning group can deliver the assignments or projects directly, as they often do via e-mail. However, confusion occurs in the email box of the instructor due to mails from many classes or from other people.

In the assignments submitted as documents, the mixing of the assignments of different student groups and the paper and printing expenses for the students can be seen as an extra expense. The main purpose of this design will be to save time for the instructor and to reduce the educational expenses of the students.

"Homework Tracking System" can actually be compared to a simple content management system. The teaching and learning group will exchange files and messages between each other and the process will run more effectively thanks to this project.

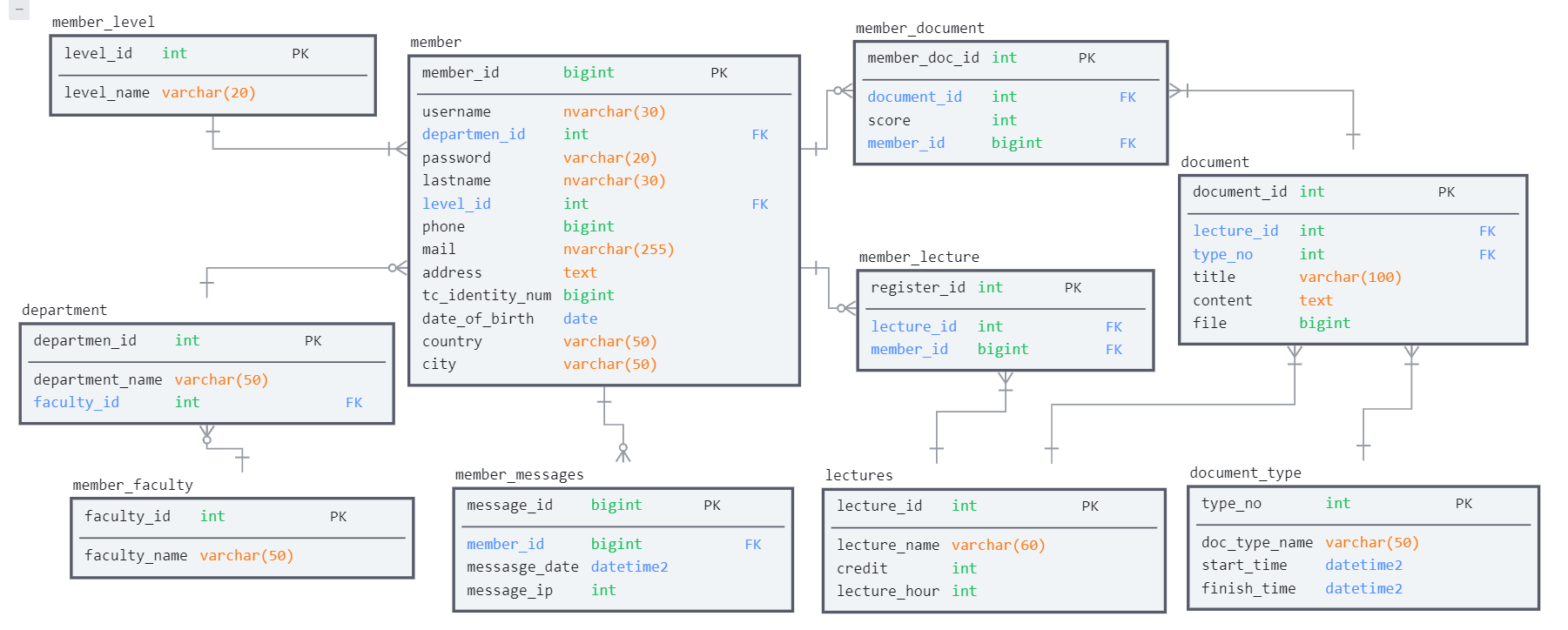
### **1.2 Potential Users of the System**

After the database design, user screen designs with different authorizations to use the system were made. Database tables were created to meet the needs of three different types of users.

* **Administrator** : The user with administrator privileges is the most authorized user with the authority to add lecturers, students, courses and announcements to the system.
* **Lecturer / Instructor** : The instructor can add his own courses and the information of the students who took this course to the system. In addition, he can write a message to his student group and follow the incoming and outgoing message boxes. The instructor delivers the files related to the assignments and projects given to the students through the system and provides the evaluation processes through the system.
* **Student** : On the student side of the process, the student can see the assignments and projects sent to him on his own screen. After the students have done the necessary work, they can send their homework or projects over the system again. The documents sent are displayed on the instructor screen. When the evaluation is made by grade, students can follow their grades on their own screens.

## **2. Design of Database System**

### **2.1 Database Design**



### **2.2 Data Dictionary**



**PHASE II**

1. Select Queries and the Outputs :

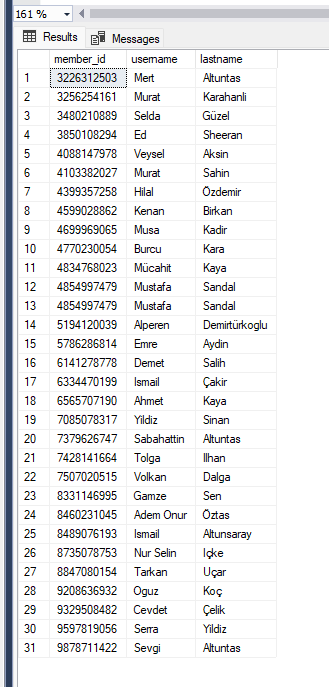
--1- ordering according to names

SELECT member\_id, username, lastname FROM member ORDER BY username ASC



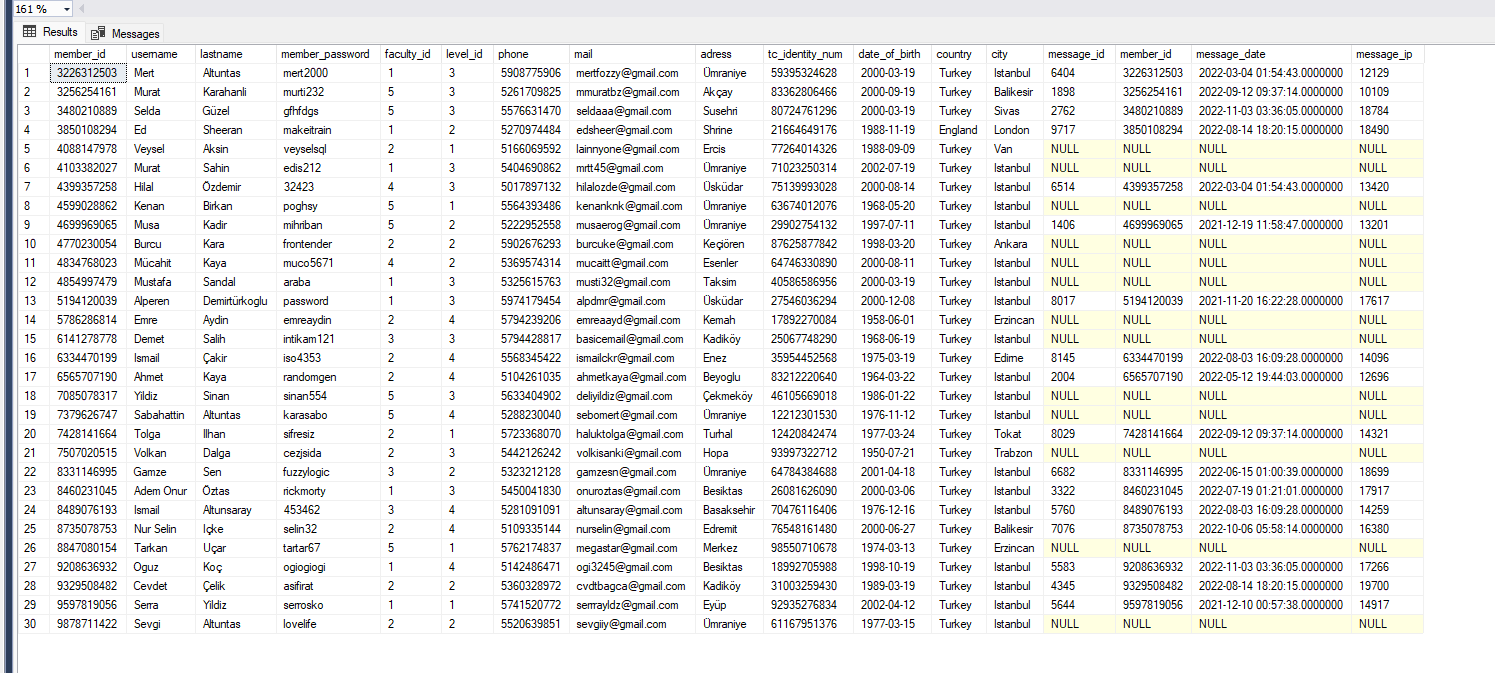
--2- left join from member lecture table (member id values)

SELECT member.member\_id, member.username, member.lastname FROM member LEFT JOIN member\_lecture ON member\_lecture.member\_id = member.member\_id



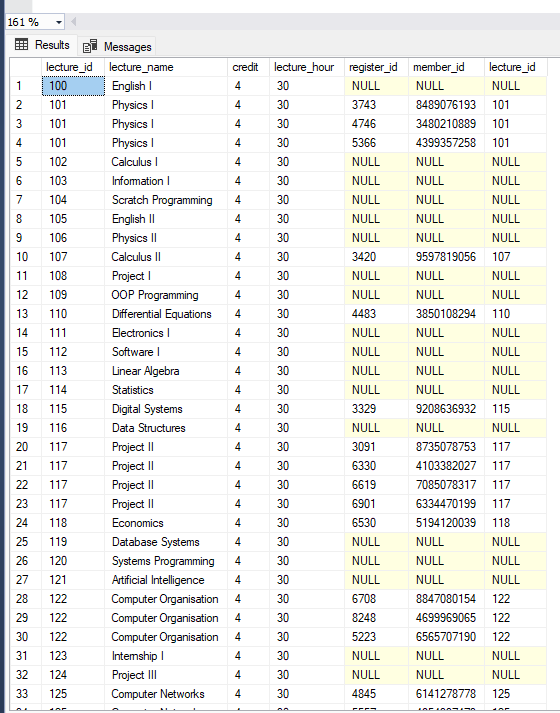
--3- full join to see member's messaging status. NULLs are not using message (nothing send or receive)

SELECT \* FROM member FULL OUTER JOIN member\_messages ON member\_messages.member\_id = member.member\_id



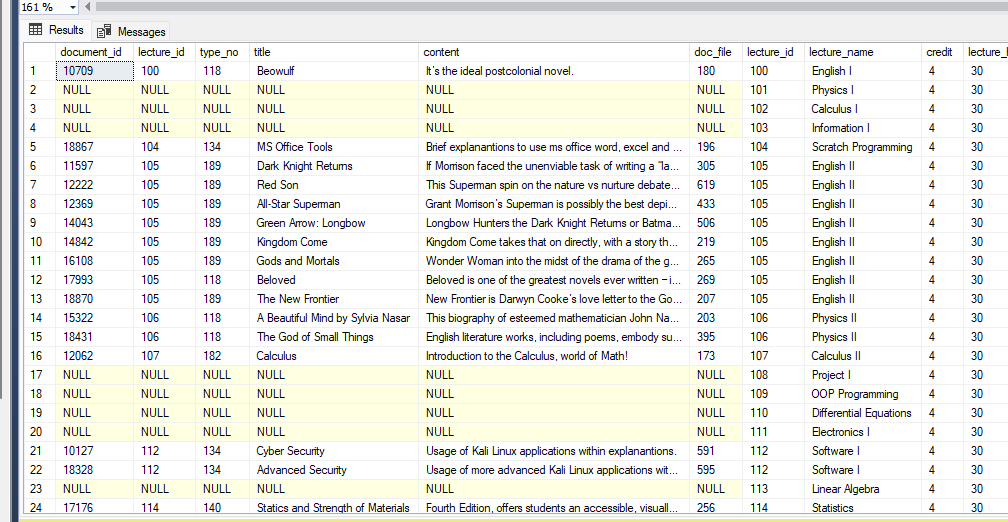
--4- full join to see which lecture have no students

SELECT \* FROM lectures FULL OUTER JOIN member\_lecture ON member\_lecture.lecture\_id = lectures.lecture\_id



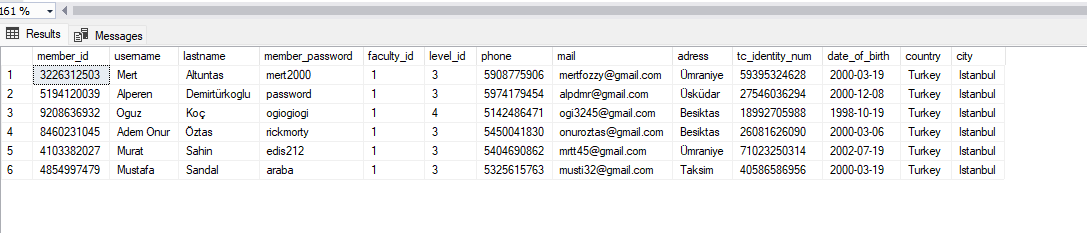
--5- right join to see which lecture have no documents (-or resources)

SELECT \* FROM document RIGHT JOIN lectures ON lectures.lecture\_id = document.lecture\_id



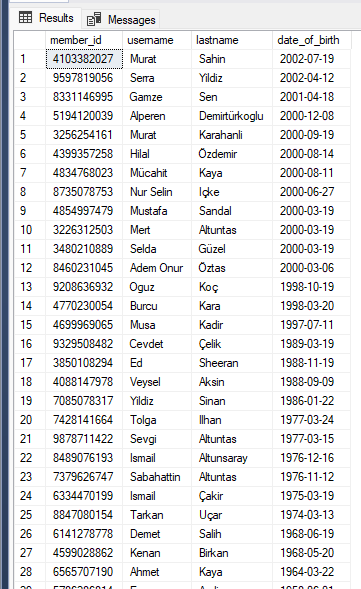
--6- ordering the student names which are going to 3th or 4th level Engineering classess. (level + faculty filter)

SELECT \* FROM member WHERE level\_id>2 AND faculty\_id=1 ORDER BY lastname, username



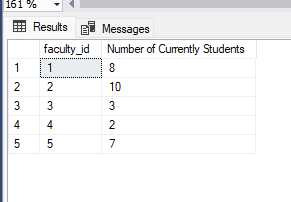
--7- ordering students according to their birthdays

SELECT member\_id, username, lastname, date\_of\_birth FROM member ORDER BY date\_of\_birth desc



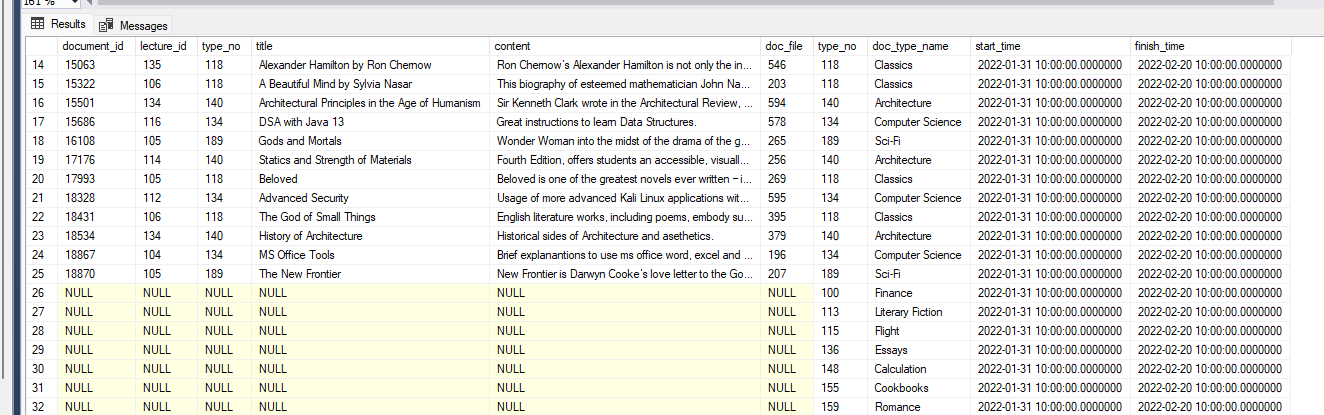
--8- grouping currently students according to their faculties (how many students continues in which faculty?)

SELECT faculty\_id, COUNT(\*) AS 'Number of Currently Students' FROM member WHERE level\_id<5 GROUP BY faculty\_id



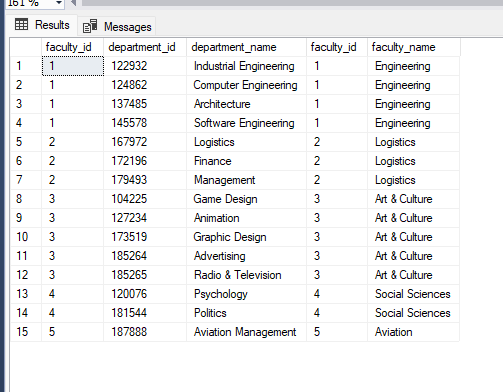
--9- full join to see which types of documents are not available

SELECT \* FROM document FULL OUTER JOIN document\_type ON document\_type.type\_no = document.type\_no



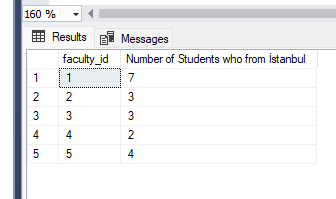
--10 right join to see the list of departments with the faculty names together

SELECT \* FROM department RIGHT JOIN member\_faculty ON member\_faculty.faculty\_id = department.faculty\_id



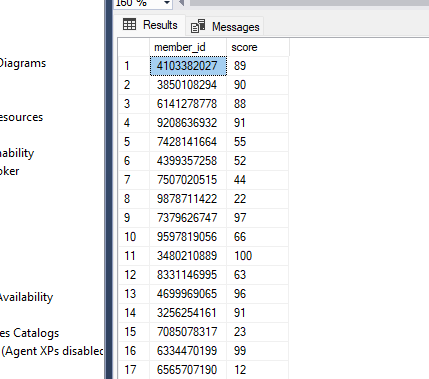
--11 grouping faculties, according to students who come from İstanbul

SELECT faculty\_id, COUNT(\*) AS 'Number of Students who from İstanbul' FROM member WHERE city like 'İ%' GROUP BY faculty\_id



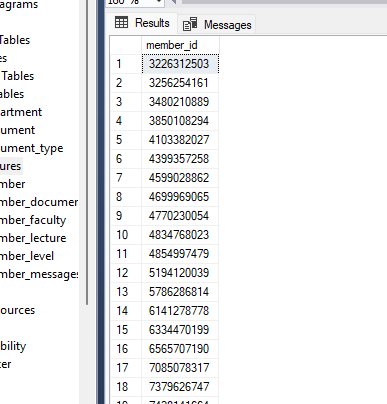
--12 join to see student's scores (id and score come from different tables)

SELECT member.member\_id, score FROM member\_document JOIN member ON member.member\_id = member\_document.member\_id



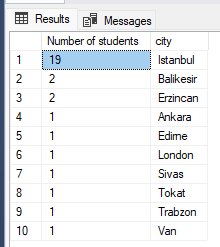
--13 intersect 2 different tables (member\_document & member\_lecture)

SELECT member\_id FROM member\_document INTERSECT select member\_id from member\_lecture ORDER BY member\_id



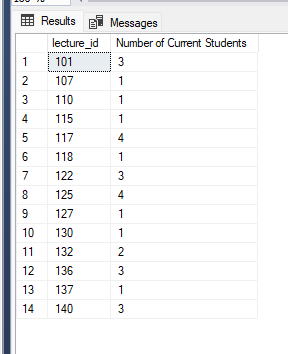
--14 grouping and ordering students according to their city

SELECT COUNT(member\_id) as 'Number of students', city FROM member GROUP BY city ORDER BY COUNT(member\_id) DESC



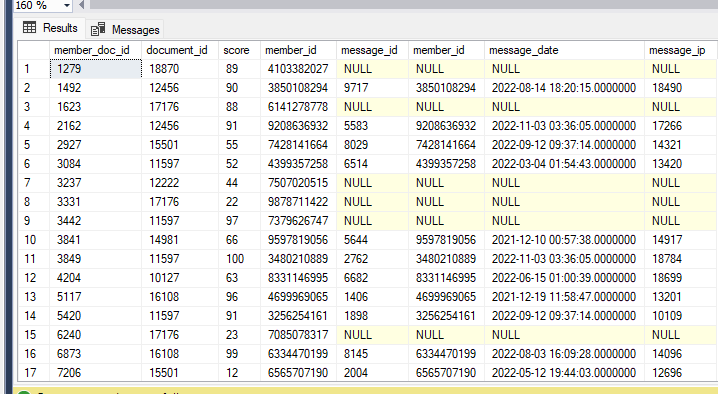
--15 joining 2 tables and grouping with lecture\_id to see "how many students taking which lecture?"

SELECT member\_lecture.lecture\_id, COUNT(member\_id) AS 'Number of Current Students' FROM member\_lecture JOIN lectures ON member\_lecture.lecture\_id=lectures.lecture\_id GROUP BY member\_lecture.lecture\_id



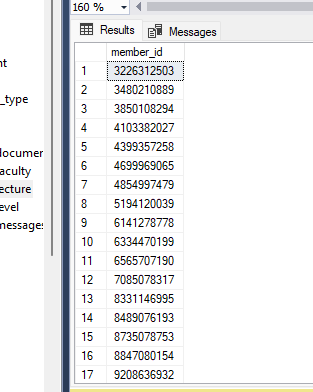
--16 left join to find, are there any messaging activity for a document

SELECT \* FROM member\_document LEFT JOIN member\_messages ON member\_messages.member\_id = member\_document.member\_id



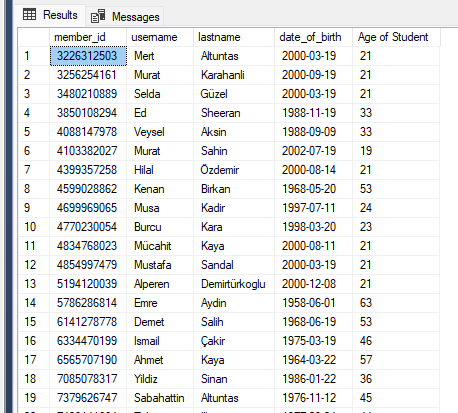
--17 a filter, finds the students who are taking a 1st or 2nd year course

SELECT member\_id from member WHERE member\_id IN (select member\_id from member\_lecture where lecture\_id < 130)



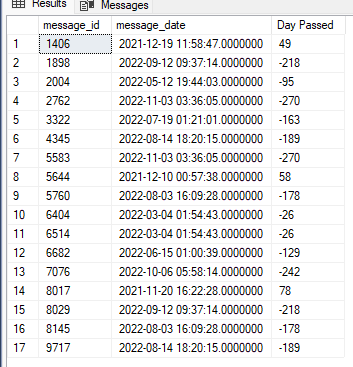
--18 datediff function to calculate student's ages from their birthday

SELECT member\_id, username, lastname, date\_of\_birth, DATEDIFF(MONTH, date\_of\_birth, GETDATE())/12 AS 'Age of Student' from member



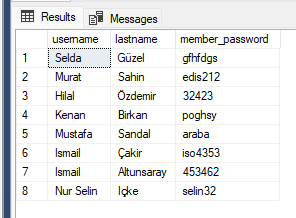
--20 calculates how many days have passed since the message was sent

SELECT message\_id, message\_date, DATEDIFF(DAY, message\_date, GETDATE()) AS 'Day Passed' from member\_messages



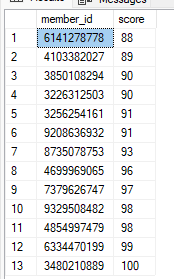
--21 finds the passwords which are not contains at least 8 characters

SELECT username, lastname, member\_password from member where len(member\_password) < 8



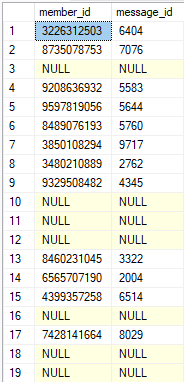
--22 finds the students who passed (higher than 70 scores) and orders by score

SELECT member\_document.member\_id, score from member\_document where score > 70 ORDER BY score



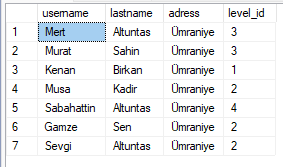
--23 right join to see students who are not messaging about any lecture

SELECT member\_messages.member\_id, message\_id from member\_messages RIGHT JOIN member\_lecture ON member\_lecture.member\_id = member\_messages.member\_id



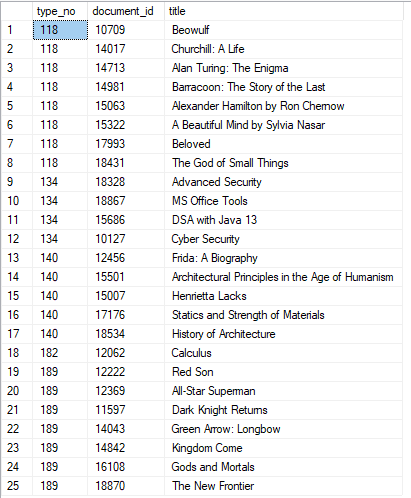
--24 left join to see levels of students who live in Ümraniye

SELECT username, lastname, adress, member.level\_id from member LEFT JOIN member\_level ON member\_level.level\_id = member.level\_id where adress like 'Ümraniye'



--25 inner join to see types of documents with their names

SELECT document.type\_no, document\_id, title from document INNER JOIN document\_type ON document\_type.type\_no = document.type\_no ORDER BY type\_no



2. Inserts, Updates and Deletes :

INSERT INTO member\_faculty (faculty\_id, faculty\_name) VALUES (5, 'Aviation')

INSERT INTO department (department\_id, department\_name, faculty\_id) VALUES (187888, 'Aviation Management', 5)

INSERT INTO lectures (lecture\_id, lecture\_name, credit, lecture\_hour) VALUES (139, 'Construction', 4, 30)

INSERT INTO document\_type (type\_no, doc\_type\_name, start\_time, finish\_time) VALUES (148, 'Autobiographies', '2022-01-31 10:00:00', '2022-02-20 10:00:00')

INSERT INTO document (document\_id, lecture\_id, type\_no, title, content, doc\_file) VALUES (15686, 116, 134, 'DSA with Java 13', 'Great instructions to learn Data Structures.', 578)

INSERT INTO member (member\_id, username, lastname, member\_password, faculty\_id, level\_id, phone, mail, adress, tc\_identity\_num, date\_of\_birth, country, city) VALUES (

3226312503, 'Mert', 'Altuntaş', 'mert2000', 1, 3, 5908775906, 'mertfozzy@gmail.com', 'Ümraniye', '59395324628', '2000-03-19', 'Turkey', 'İstanbul')

--UPDATES

UPDATE member SET member\_password = 'admin123' where member\_id = 3226312503

UPDATE document SET title = 'The Great Gatsby' where document\_id = 10709

UPDATE lectures SET credit = 5 where lecture\_id between 120 and 130

UPDATE document\_type SET finish\_time = '2022-03-19 21:00:00' where doc\_type\_name = 'Finance'

UPDATE member SET faculty\_id = 3 where phone = 5520639851

--DELETES

DELETE FROM member\_messages WHERE message\_id = 9717

DELETE FROM document\_type WHERE type\_no = 174

3. Alter Tables and Drops :

--ALTER & DROPS

ALTER TABLE document\_type ADD Related\_Lecture nvarchar(60)

ALTER TABLE document ADD author\_name nvarchar(60)

ALTER TABLE lectures ADD lecturer\_name nvarchar(60)

ALTER TABLE member\_faculty ADD faculty\_location text

ALTER TABLE member ADD department\_id int

ALTER TABLE document\_type DROP COLUMN Related\_Lecture

ALTER TABLE document DROP COLUMN author\_name

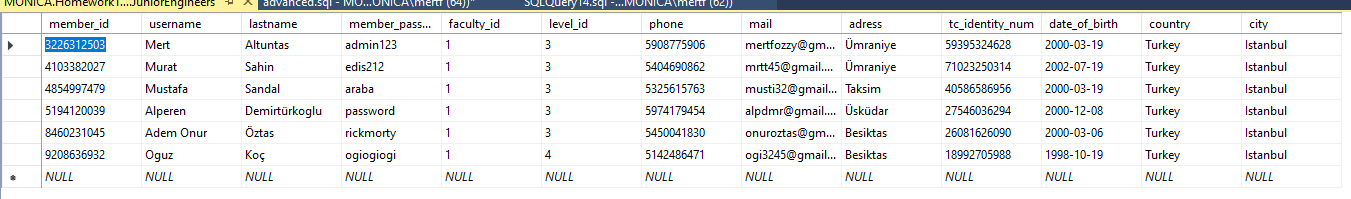
ALTER TABLE lectures DROP COLUMN lecturer\_name

ALTER TABLE member\_faculty DROP COLUMN faculty\_location

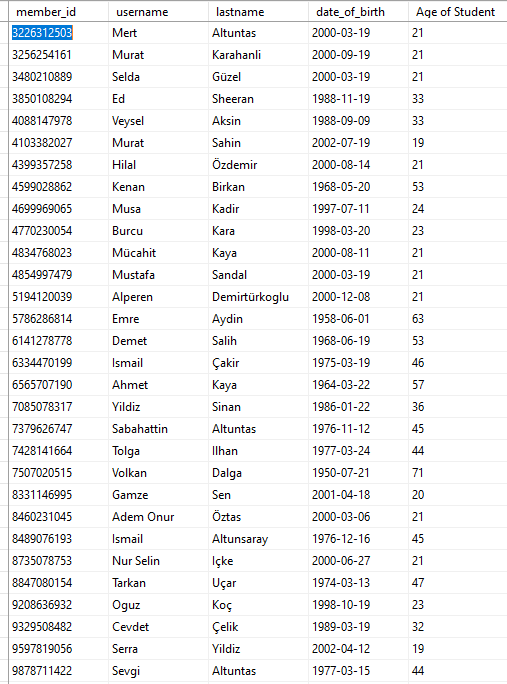
ALTER TABLE member DROP COLUMN department\_id

4. Create Views :

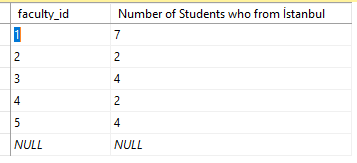
CREATE VIEW JuniorEngineers AS SELECT \* FROM member WHERE level\_id>2 AND faculty\_id=1



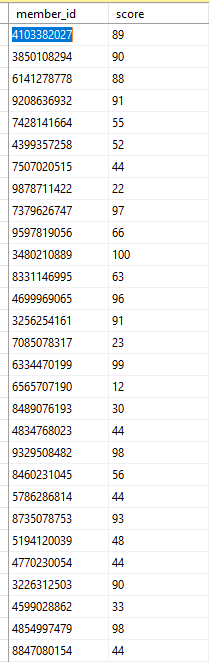
CREATE VIEW Birthdays AS SELECT member\_id, username, lastname, date\_of\_birth, DATEDIFF(MONTH, date\_of\_birth, GETDATE())/12 AS 'Age of Student' from member



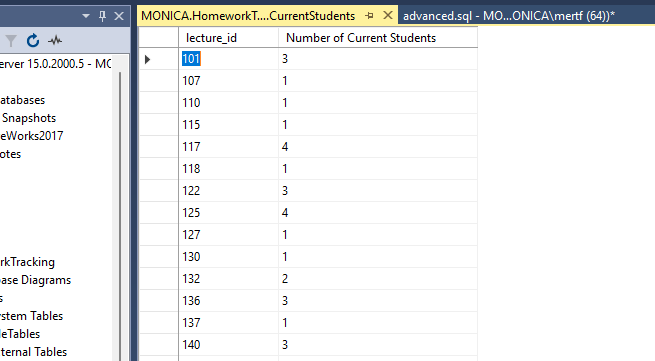
CREATE VIEW IstanbulStudents AS SELECT faculty\_id, COUNT(\*) AS 'Number of Students who from İstanbul' FROM member WHERE city like 'İ%' GROUP BY faculty\_id



CREATE VIEW Grades AS SELECT member.member\_id, score FROM member\_document JOIN member ON member.member\_id = member\_document.member\_id



CREATE VIEW CurrentStudents AS SELECT member\_lecture.lecture\_id, COUNT(member\_id) AS 'Number of Current Students' FROM member\_lecture JOIN lectures ON member\_lecture.lecture\_id=lectures.lecture\_id GROUP BY member\_lecture.lecture\_id



5. Indexes :

--INDEXES

CREATE INDEX inx\_username ON member (username, lastname)

CREATE UNIQUE INDEX unique\_id ON document (document\_id)

6. Triggers and Store Procedures :

CREATE TRIGGER NewStudentRegister ON member

AFTER INSERT

AS

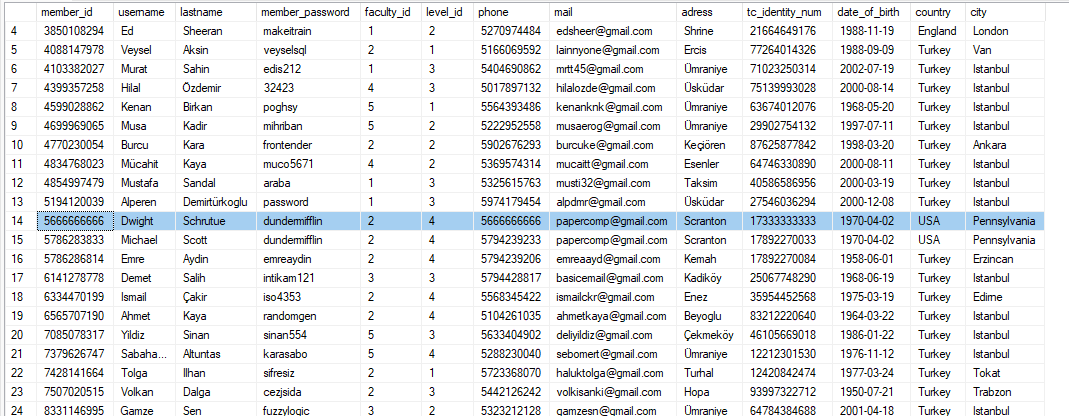
BEGIN

select \* from member

END

INSERT INTO member (member\_id, username, lastname, member\_password, faculty\_id, level\_id, phone, mail, adress, tc\_identity\_num, date\_of\_birth, country, city) VALUES (

5666666666, 'Dwight', 'Schrutue', 'dundermifflin', 2, 4, 5666666666, 'papercomp@gmail.com', 'Scranton', '17333333333', '1970-04-02', 'USA', 'Pennsylvania')



CREATE TRIGGER DeleteBook on Document

AFTER DELETE

AS

BEGIN

IF(EXISTS(select \* from deleted where type\_no = 140))

BEGIN

RAISERROR('Architecture books cannot deleted because of University copyrights.',0,0)

ROLLBACK TRANSACTION

END

END

DELETE FROM document where document\_id = 15007

