Ege Üniversitesi

Bilgisayar Mühendisliği Bölümü

DATABASE MANAGEMENT

2021-2022 FALL

TERM PROJECT

Delivery Date: 02/02/2022

Selin PAKSOY 05190000029

Mehmet Gökberk Ayhan 05190000098

Contents

LINKEDIN ANALYSIS

1) LinkedIn is a business- and employment-oriented online service that operates via websites and mobile apps which is launched on May 5, 2003. Firstly, the platform is used for professional networking and career development. Now, LinkedIn is a social networking website for companies and people. The purpose of LinkedIn is to allow people and companies to network with each other. LinkedIn lets people prepare a profile page where they can share their education, skills, hobbies, and work experiences with other people. On LinkedIn, people can follow other people to share their experiences or companies to stay up to date with job offers. Users can share, like, save or comment on a post. Also, LinkedIn can be used to create or join groups, send messages, etc.

2)

2a.

It allows users to expand their networks in the business world by introducing themselves appropriately. So that the user, if he is looking for a job, can examine the job offers suitable for him and view the status of his other connections. If the account is a company account, this company can find the employees they are looking for easier and faster thanks to LinkedIn. In addition, both types of users can share posts, communicate with other accounts, so LinkedIn offers us a professional social media opportunity.

2b.

Key Entities:

Users -> User must be a company or a person account

Company -> Company account where people work and can offer people an optional job

Person -> Person accounts that have worked or are working somewhere, have hobbies, skills, certificates; graduated from a school, etc.

Posts –> shared, liked, saved, commented by the user

Forum-> where posts are shared

Job Alert- offer that companies can create and that can be accepted by the user

Experience -> shows the experience of employees in companies

Group -> users can create a new group or join already created groups

Skill -> users can have many skills

Hobby -> users may have multiple hobbies

Certificate-> The user may have gained a certificate because of various evaluations.

c. What are the characteristics of each entity?

USERS

- **E-mail**
- **UserID**
- **Password**
- **Phone**
- **Address**

The user can be a person, a company:

PERSON

- Person ID -PK
- Person’s Name (FName, Minit, LName)
- Date of Birth
- Sex
- CV

COMPANY

- CompanyName
- EmployeeNumber

POST

The users can post something with others, or like or comment on others’ posts.

- Post-ID
- ContentType
- PostedDate

GROUP

The user can create a group or join another group.

- G-ID (Group-ID)
- GroupName
- MemberNo

COURSE

The user can take a course or give a course to another people.

- CCode (CourseCode)
- CourName(CourseName)
- Level

CERTIFICATE

The user can earn certificate.

- Cer-ID (Certificate ID)
- CourseID
- UserID
- CerDate(Certificate Date)
- CerName(Certificate Name)

SKILL

The user can have skills, skills have id and name.

- SkID(Skill ID)
- SName(Skill Name)

HOBBY

The user also can have hobbies.

- H-ID(HobbyID)
- HName(HobbyName)

d. What relationships exists among the entities?

Each account must have a unique id, unique e-mail, password, phone number, and address. Each account must belong to a person or a company. If the user is a person, it should have a name, CV, date of birth, sex. If it is a company, the name of the company and the number of employees should be recorded. Users can follow other users or send messages to other users. The type of messages sent the content of the message, and the time of sending are kept. Users can share posts with other users or like, save or comment on posts. Each post has a unique ID, content, and date of sharing. Companies can provide experience for persons. Experience must be provided by companies. The experience has a unique id, location, description, title, and position where the person works. Persons can gain experience in companies. Companies can create job advertisement and people can see this advertisement. Job advertisement should have a unique ID, title, description, position for the job, and location. Users can create as many groups as they want. Groups must be created by one person, and since the creator will be in the group, they must be a member of the group. The group has its unique ID, name, and number of members. Persons can give as many courses as they want, but a course must be given by one person. Anyone can take a course, but even if the course is not taken by anyone. The course must have a unique code, name, and level of the course. A person can have more than one certificate, more than one person can receive this certificate. Each certificate must have a unique ID, name, given date, the ID of the given course and the ID of the person who received the certificate. A person can have more than one skill, a skill can be owned by more than one person. Each skill must have a unique ID and name. A person can have more than one hobby, other people can have the same hobby. Each hobby should have a unique ID and name.

The user must have a company or person type. Companies can *create* job offers and people can *view* this offer. Users can *share*, *like*, *save* or *comment* as many posts as they want. Users can follow or send to messages to other users. Also, persons can *create* a group or can be *member of* others' groups. Person also can *have* certificate, skills, or hobbies. Person can have *experience* with companies.

COMPANY 1 CREATE N EXPERIENCE

PERSON M VIEW N EXPERIENCE

COMPANY 1 CREATE N JOB_AD

JOB_AD N VIEW M PERSON

PERSON 1 CREATE N GROUP

GROUP M MEMBER_OF N PERSON

PERSON N TAKES M COURSE

COURSE N GIVES 1 PERSON

PERSON N HAS M CERTIFICATE

PERSON N SKILLS M SKILL

USER N MESSAGES M

USER N FOLLOWS M

USER N LIKES M POST

USER 1 POSTS N POST

USER N COMMENTS M POST

USER N SAVES M POST

e. What are the constraints related to entities, their characteristics and the relationships among them?

CONSTRAINTS:

GRUP:

CONSTRAINT pk_GRUP primary key (GroupID), CONSTRAINT fk_GRUP_PERSON foreign key (CreatorPersonUserID) references PERSON(PersonUserID)

MEMBER_OF:

CONSTRAINT pk_MEMBER_OF primary key (MemberPersonUserID, MemberGroupID),

CONSTRAINT fk_MEMBER_OF_PERSON foreign key (MemberPersonUserID) references PERSON(PersonUserID),

CONSTRAINT fk_MEMBER_OF_GRUP foreign key (MemberGroupID) references GRUP(GroupID)

COMPANY:

CONSTRAINT uq_CompanyName UNIQUE (CompanyName),

CONSTRAINT pk_COMPANY primary key (CompanyUserID),

CONSTRAINT fk_COMPANY_USERS foreign key (CompanyUserID) references USERS(UserID)

EXPERIENCE:

CONSTRAINT pk_EXPERIENCE primary key (Company, ExperienceUserID), CONSTRAINT fk_EXPERIENCE_COMPANY foreign key (Company) references COMPANY(CompanyUserID),

CONSTRAINT fk_EXPERIENCE_PERSON foreign key (ExperienceUserID) references PERSON(PersonUserID)

JOB_ALERT:

CONSTRAINT pk_JOB_ALERT primary key (AlertID, AlertingCompanyUserID),

CONSTRAINT fk_JOB_ALERT_COMPANY foreign key (AlertingCompanyUserID) references COMPANY(CompanyUserID)

VIEW_ALERT:

CONSTRAINT pk_VIEW_ALERT primary key (ViewingPersonID, ViewedAlertID),

CONSTRAINT fk_VIEW_ALERT_PERSON foreign key (ViewingPersonID) references PERSON(PersonUserID),

CONSTRAINT fk_VIEW_ALERT_JOB_ALERT foreign key (ViewedAlertID) references JOB_ALERT(AlertID)

FORUM:

CONSTRAINT pk_FORUM primary key (ForumTitle),

CONSTRAINT fk_FORUM_USERS foreign key (CreatorUserID) references USERS(UserID)

POST:

CONSTRAINT pk_POST primary key (PostID),

CONSTRAINT fk_POST_FORUM foreign key (ForumTitle) references FORUM(ForumTitle),

CONSTRAINT fk_POST_USERS foreign key (PosterUserID) references USERS(UserID)

COMMENTS:

CONSTRAINT pk_COMMENTS primary key (CommentingUserID, CommentedPostID),

CONSTRAINT fk_COMMENTS_USERS foreign key (CommentingUserID) references USERS(UserID)

SAVES:

CONSTRAINT pk_SAVES primary key (SavingUserID, SavedPostID),

CONSTRAINT fk_SAVES_USERS foreign key (SavingUserID) references USERS(UserID)

LIKES:

CONSTRAINT pk_LIKES primary key (LikingUserID, LikedPostID),

CONSTRAINT fk_LIKES_USERS foreign key (LikingUserID) references USERS(UserID)

SKILL:

CONSTRAINT pk_SKILL primary key (SkillID),

CONSTRAINT fk_SKILL_USERS foreign key (SkillPersonUserID) references PERSON(PersonUserID)

HOBBY:

CONSTRAINT pk_HOBBY primary key (HobbyID),

CONSTRAINT fk_Hobby_USERS foreign key (HobbyPersonUserID) references PERSON(PersonUserID)

HAS:

CONSTRAINT pk_HAS primary key (HobbysPersonUserID, HHobbyID),

CONSTRAINT fk_HAS_PERSON foreign key (HobbysPersonUserID) references PERSON(PersonUserID),

CONSTRAINT fk_HAS_HOBBY foreign key (HHobbyID) references HOBBY(HobbyID)

MOODLE ANALYSIS

1-Write a brief explanation using your own words (in English) about these applications in terms of their scope::

Moodle (stands for, *modular object-oriented dynamic learning environment)* is a free open-source education management system widely used in distance education and online teaching systems. It was first developed by Martin Dougiamas, to help educators create and manage online courses, 19 years ago. Moodle has course specific customizable management features. Educators can modify their websites with variety of plugins and customization settings -such as themes- available. There are several roles of individuals can be in a Moodle course where each role provides access to a set of capabilities. We will be explaining them on later parts. Basically, system works between teachers-course assistants-students. On their site, teachers can post, make announcements, see students enrolled in the course, assign assignments, and rate them. C*ourse assistan*t is a special role that can only be assigned to an already-enrolled user from higher hierarchy. Course assistants can access the assignments uploaded by students and can grade them, but they don’t have access to the gradebook and they can’t share posts like teachers. Students can participate in course activities and can download course material that are shared by teachers. Activities like assignments are submitted by one student only. This is the main logic behind the Moodle.

- **Site Administrator**
  - **This type of user can do anything.**
- **Manager**
  - **Is similar to site administrator but with a few tweaks**
- **Course Creator**
  - **This user creates courses. By default a course creator is the Teacher.**
- **Teacher**
  - **Can manage and edit the contents of courses**
- **Non-Editing Teacher**
  - **This type of user can grade courses but can’t do much else. Sort of like course assistans.**
- **Student**
  - **Students can access and participate in courses and performsa assigned tasks.**
- **Guest**
  - **Guests can view courses but they can’t be a part of them**
- **Authenticated User**
  - **This is a meta role. All logged in users have this role.**

2. Write an analysis report for each web application: a. What is the aim of each application? b. What are the main entities of them? c. What are the characteristics of each entity? d. What relationships exists among the entities? e. What are the constraints related to entities, their characteristics, and the relationships among them?

2/A:

Moodle is an online education platform that is designed for educators and learners, where you can build customizable learning environments, assign tasks, share data, participate in events and courses. It’s designed for both teaching and learning. This means that Moodle delivers a set of learner-centric tools and collaborative learning environments that empower both teaching and learning. It’s also very easy to use with simple interface and drag-drop features so any age of user with any experience can easily use this application.

2/B:

In actual Moodle database there are way more entities. But in this project, we will be focusing on these key entities:

PERSON

  - A person is the main user of this system.
- **ACCOUNT**
  - Every person has some kind of account.
- **COURSE**
  - This is where classes happen.
- **CONTENT**
  - Any informational material that is needed for participation or understanding classes.
- **QUIZ**
  - A test of knowledge. Series of questions that is asked to students.
- **ASSIGNMENT**
  - Long term homework given to students.
- **QUESTIONS**
  - Contents of a quiz or an assignment.
- **CERTIFICATE**
  - A certificate of achievement that you get for successfully completing all the tasks.
- **MESSAGAES**
  - Communication box between two people. (a person can send and receive messages- unary relationship)
- **FEEDBACK**
  - Students can send feedback to teachers about things that they don't like.

2/C:

PERSON:

A person is the main user of this system. This person can be a teacher, student, assistant or administrator. Apart from the differences, each person has their username (first name, middle name initials, last name), date of birth, address, phone number, e-mail and a unique user id stored in this database.

- **Teacher:**
  If this person is a teacher, their expertness, department ,salary and diploma information (diploma given date, college obtained) is stored.
- **Student:**

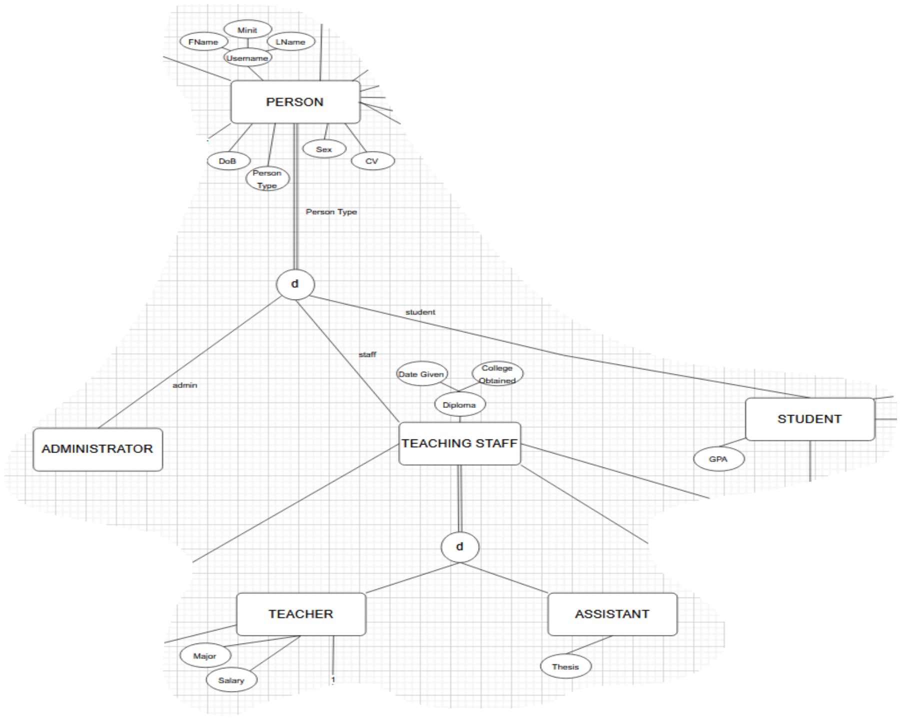
  If this person is a student, their gpa, enrolled courses and their department is stored.

- **Assistant:**

  If this person is an assistant, their thesis and diploma information is stored.

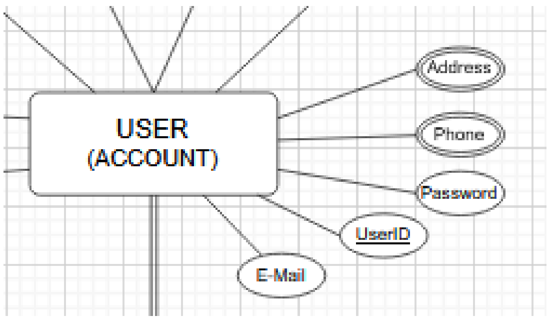
- **Administrator:**

  If this person is an administrator, there is no extra information is stored.



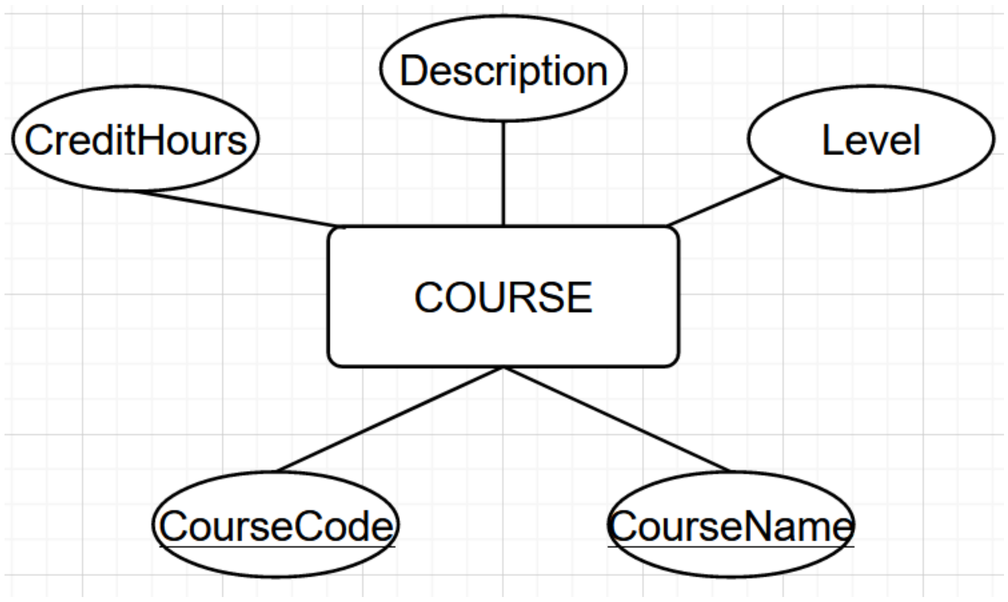
ACCOUNT:

Every user listed above must have some kind of an account. And every account has their person’s role and a unique account id stored in them.



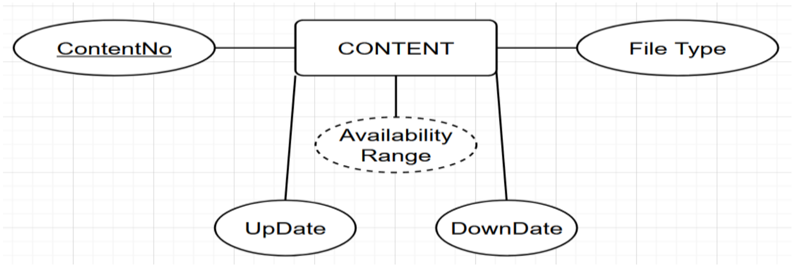
COURSE:

Every course has their credit hours, level of education, description and unique course code and course name kept in this database.



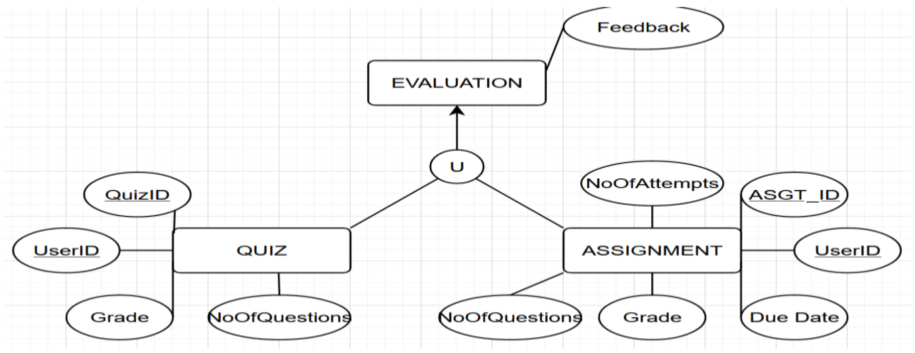
CONTENT:

Materials that are used in the courses are called, course contents. It can be of different file types. A content has a file type, availability date range (upload date and self-destruct date), and a unique content no.



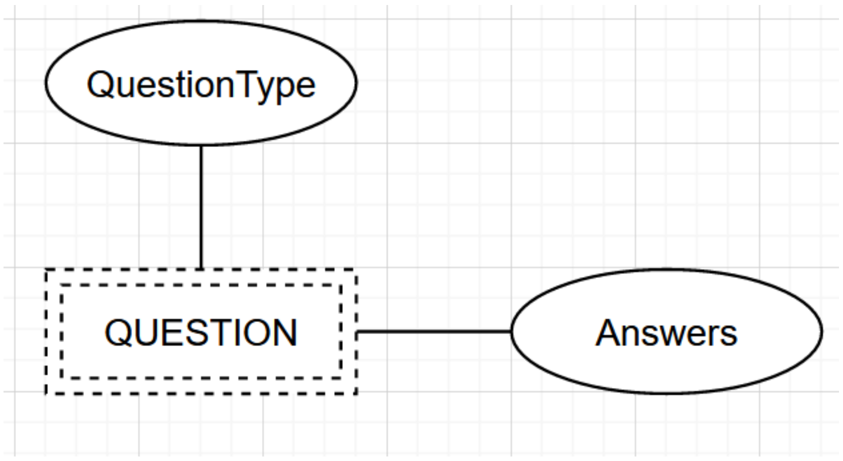
EVALUATION:

Evaluation is done by QUIZ and ASSIGNMENTS. Every Quiz is graded, has a number of questions and has it’s unique quiz no. Every Assignment is also graded and has a number of questions. They also have a user’s number of attempts, due date and a unique assignment id. And Evaluation has a feedback process as well.



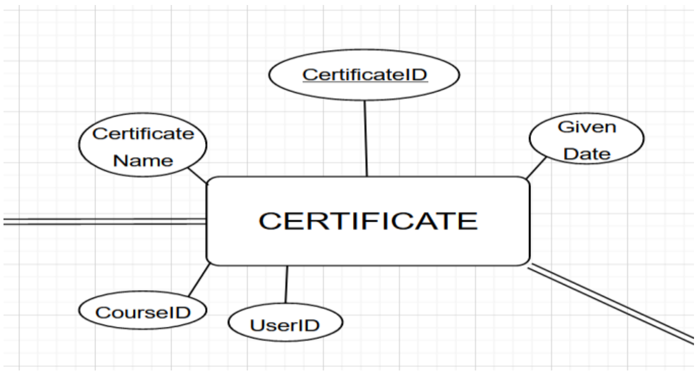
QUESTIONS:

Every Evaluation process is made out of questions. Every question entity has a question type.



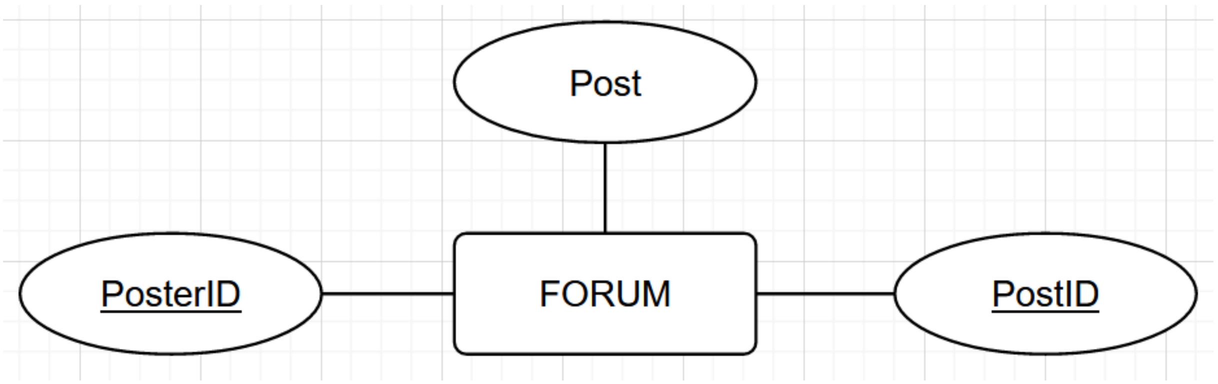
CERTIFICATE:

A certificate is given after a student achieves necessary amount achievements for a course. A certificate has given date, course name and a unique certificate id.



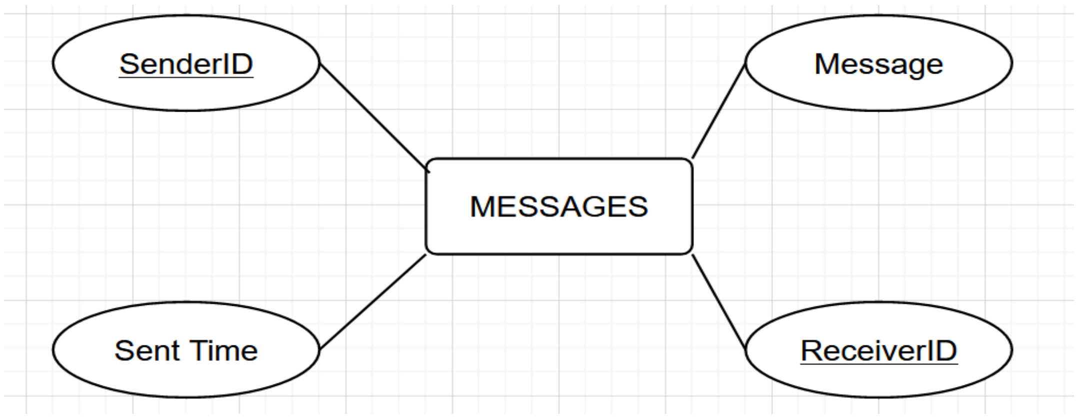
FORUM:

Users share their ideas and ask and answer each other's posts. Every user has to post with their user id and every post has it's unique post id.



MESSAGES:

It’s a communication process between two people. A person can both send and receive messages. Every message has information about both sender and receiver and time sent.



2/D

A person must have an account and an account must be bound to one person only. A person can message with another person as well. A person can share posts in forums and forums may have multiple people posting on them. If this person is a teacher, they may be giving more than one course, but a course must have only one teacher. A teacher and assistants can upload contents to courses, naturally they can upload more than one content per course. Students can use these contents via downloading them. Accounts earn certificates. A certificate may be possessed by more than one person and a person may achieve more than one certificate. Administrators provide support and they contact with other group of people.

Teachers will make evaluations for students. So, a course must have at least one of these evaluation tasks. And students are evaluated. Quizzes and assignments can be graded by both teacher and assistants. Both of these evaluation elements are made of multiple questions. These evaluation tasks are answered by students. A student may have more than one evaluation per course. Evaluations grant students, certificates. After the exams are done, students get to give feedback about them. A teacher will receive these feedbacks.

PERSON	1	HAS	1	ACCOUNT
PERSON	M	EARNS	M	CERTIFICATE
PERSON	1	MESSAGES	1	PERSON
PERSON	M	POSTS ON	M	FORUM
TEACHER	1	GIVES	M	COURSE
TEACHER	1	DOES	M	EVALUATIONS
TEACHER	1	UPLOADS	M	CONTENT
TEACHER	1	GRADES	M	EVALUATIONS
TEACHER	1	RECIEVES	M	FEEDBACK

ASSISTANT	1	GRADES	M	EVALUATIONS
ASSISTANT	1	UPLOADS	M	CONTENT
STUDENTS	1	TAKE	M	EVALUATIONS
STUDENTS	M	USE	M	CONTENT
STUDENTS	M	ENROLL TO	M	COURSE
STUDENTS	1	GIVE	M	FEEDBACK
ADMINISTRATORS	M	SUPPORT	M	PERSON ?????
EVALUATIONS	M	GRANTS	1	CERTIFICATE
EVALUATIONS	M	HAS	M	QUESTIONS
COURSE		CONTAINS		CONTENT
COURSE		HAS		EVALUATIONS

2/E

Constraints:

Users:

CONSTRAINT pk_USERS primary key (UserID),

CONSTRAINT uq_EMail UNIQUE (EMail)

PhoneNumber:

CONSTRAINT pk_USER_PHONE primary key (PhoneUsersID, PhoneNumber),

CONSTRAINT pf_USER_PHONE_USERS foreign key (PhoneUsersID) references USERS(UserID)

Address:

CONSTRAINT pk_USER_ADDRESS primary key (AddressUsersID, Address),

CONSTRAINT pf_USER_ADDRESS_USERS foreign key (AddressUsersID) references USERS(UserID)

Person:

CONSTRAINT pk_PERSON primary key (PersonUserID),

CONSTRAINT fk_PERSON_USERS foreign key (PersonUserID) references USERS(UserID)

Messages:

CONSTRAINT pk_MESSAGES primary key (SenderUserID, RecieverUserID),

CONSTRAINT fk_MESSAGES_PERSON1 foreign key (SenderUserID) references PERSON(PersonUserID),

CONSTRAINT fk_MESSAGES_PERSON2 foreign key (RecieverUserID) references PERSON(PersonUserID)

Administrator:

CONSTRAINT pk_ADMINISTRATOR primary key (AdminID),

CONSTRAINT fk_ADMINISTRATOR_USERS foreign key (AdminID) references PERSON(PersonUserID)

Teaching Staff:

CONSTRAINT pk_TEACHING_STAFF primary key (TeachingUserID),

CONSTRAINT fk_TEACHING_STAFF_USER foreign key (TeachingUserID) references PERSON(PersonUserID)

Student:

CONSTRAINT pk_STUDENT primary key (StudentUserID),

CONSTRAINT fk_STUDENT_USERS foreign key (StudentUserID) references Person(PersonUserID)

Teacher:

CONSTRAINT pk_TEACHER primary key (TeacherUserID),

CONSTRAINT fk_TEACHER_USER foreign key (TeacherUserID) references TEACHING_STAFF(TeachingUserID)

Feedback:

CONSTRAINT pk_FEEDBACK primary key (FeedbackID),

CONSTRAINT fk_FEEDBACK_STUDENT foreign key (SendingStudentUserID) references STUDENT(StudentUserID),

CONSTRAINT fk_FEEDBACK_TEACHER foreign key (RecievingTeacherUserID) references TEACHER(TeacherUserID)

Assistant:

CONSTRAINT pk_ASSISTANT primary key (AssistantUserID), CONSTRAINT fk_ASSISTANT_USERS foreign key (AssistantUserID) references TEACHING_STAFF(TeachingUserID)

Course:

CONSTRAINT pk_COURSE primary key (CourseCode),

CONSTRAINT fk_COURSE foreign key (CourseTeacherUserID) references TEACHER(TeacherUserID)

Content:

CONSTRAINT pk_CONTENT primary key (ContentNo),

CONSTRAINT fk_CONTENT_TEACHING_STAFF foreign key (UploaderUserID) references TEACHING_STAFF(TeachingUserID)

Uploads:

CONSTRAINT pk_UPLOADS primary key (UppingTeachingStaffUserID),

CONSTRAINT fk_UPLOADS_TEACHING_STAFF foreign key (UppedContentNo) references TEACHING_STAFF(TeachingUserID)

Evaluation:

CONSTRAINT pk_EVALUATION primary key (EvaluationID, TakingStudentUserID, MakerUserID, CourseID, GraderUserID),

CONSTRAINT fk_EVALUATION_TEACHING_STAFF1 foreign key (MakerUserID) references TEACHING_STAFF(TeachingUserID),

CONSTRAINT fk_EVALUATION_TEACHING_STAFF2 foreign key (GraderUserID) references TEACHING_STAFF(TeachingUserID),

CONSTRAINT fk_EVALUATION_COURSE foreign key (CourseID) references COURSE(CourseCode),

CONSTRAINT fk_EVALUATION_STUDENT foreign key (TakingStudentUserID) references STUDENT(StudentUserID)

Assignment:

CONSTRAINT pk_ASSIGNMENT primary key (ATakingStudentUserID, AEvaluationID),

CONSTRAINT fk_ASSIGNMENT_STUDENT foreign key (ATakingStudentUserID) references STUDENT(StudentUserID),

CONSTRAINT fk_ASSIGNMENT_EVALUATION foreign key (AEvaluationID) references EVALUATION(EvaluationID)

Quiz:

CONSTRAINT pk_QUIZ primary key (QTakingStudentUserID, QEvaluationID),

CONSTRAINT fk_QUIZ_STUDENT foreign key (QTakingStudentUserID) references STUDENT(StudentUserID),

CONSTRAINT fk_QUIZ_EVALUATION foreign key (QEvaluationID) references EVALUATION(EvaluationID)

Enrolls:

CONSTRAINT pk_ENROLLS primary key (EnrolledCourseCode, EnrollingStudentUserID),

CONSTRAINT fk_ENROLLS_COURSE foreign key (EnrolledCourseCode) references COURSE(CourseCode),

CONSTRAINT fk_ENROLLS_STUDENT foreign key (EnrollingStudentUserID) references STUDENT(StudentUserID)

Take:

CONSTRAINT pk_TAKE primary key (TStudentUserID, TEvaluationID, EMakerID, EGraderID, EvalTakenCourseID),

CONSTRAINT fk_TAKE_STUDENT foreign key (TStudentUserID) references STUDENT(StudentUserID),

CONSTRAINT fk_TAKE_EVALUATION foreign key (TEvaluationID) references EVALUATION(EvaluationID),

CONSTRAINT fk_TAKE_TEACHING_STAFF1 foreign key (EMakerID) references TEACHING_STAFF(TeachingUserID),

CONSTRAINT fk_TAKE_TEACHING_STAFF2 foreign key (EGraderID) references TEACHING_STAFF(TeachingUserID),

CONSTRAINT fk_TAKE_COURSE foreign key (EvalTakenCourseID) references COURSE(CourseCode)

Certificate:

CONSTRAINT pk_CERTIFICATE primary key (CertificateID),

CONSTRAINT fk_CERTIFICATE_COURSE foreign key (CertificateCourseID) references COURSE(CourseCode),

CONSTRAINT fk_CERTIFICATE_STUDENT foreign key (CertificatedStudentUserID) references STUDENT(StudentUserID),

CONSTRAINT fk_CERTIFICATE_EVALUATION foreign key (CertificatingEvaluationID) references EVALUATION(EvaluationID), CONSTRAINT fk_CERTIFICATE_TEACHING_STAFF1 foreign key (EMakerID) references TEACHING_STAFF(TeachingUserID),

CONSTRAINT fk_CERTIFICATE_TEACHING_STAFF2 foreign key (EGraderID) references TEACHING_STAFF(TeachingUserID)

Earns:

CONSTRAINT pk_EARNS primary key (EarningPersonUserID, EarnedCertificateID),

CONSTRAINT fk_EARNS_PERSON foreign key (EarningPersonUserID) references PERSON(PersonUserID),

CONSTRAINT fk_EARNS_CERTIFICATE foreign key (EarnedCertificateID) references CERTIFICATE(CertificateID)

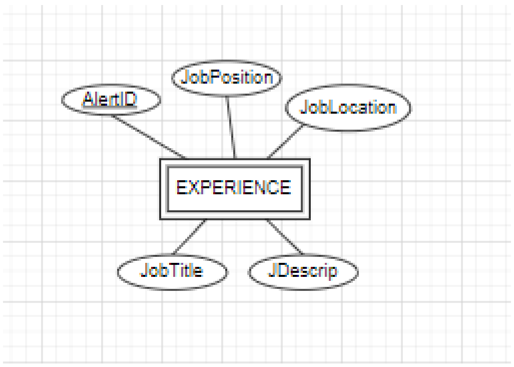
Questions:

CONSTRAINT pk_QUESTIONS primary key (QuestionID, ExamEvaluationID),

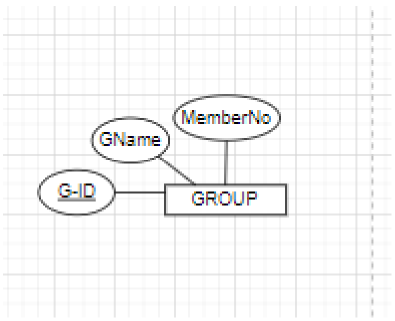
CONSTRAINT fk_QUESTIONS_EVALUATION foreign key (ExamEvaluationID) references EVALUATION(EvaluationID)

LinkedIn Design:

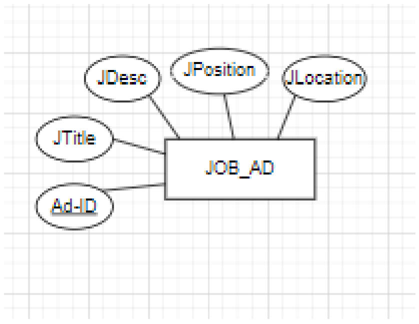
EXPERIENCE:



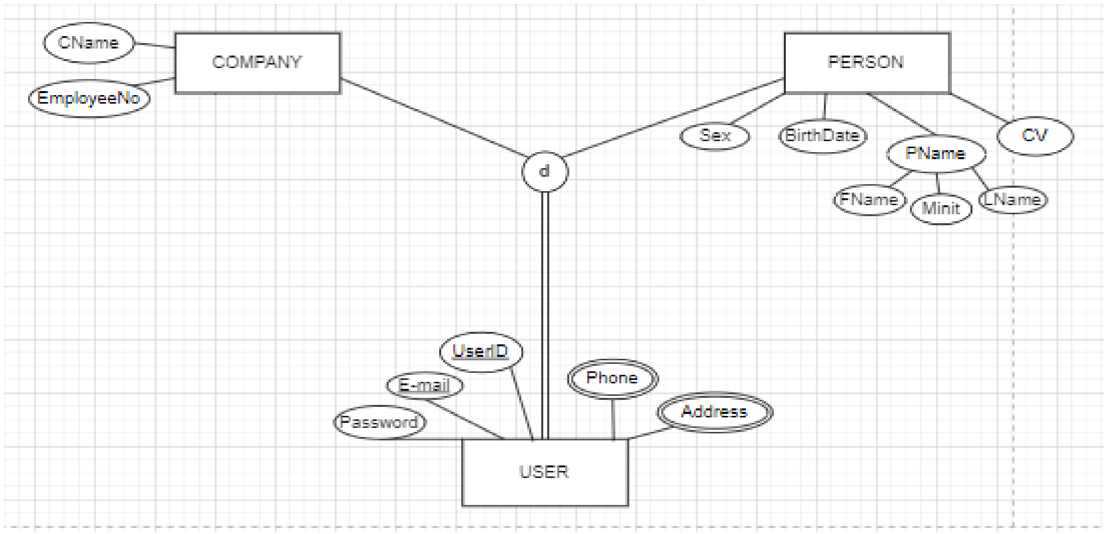
GROUP:



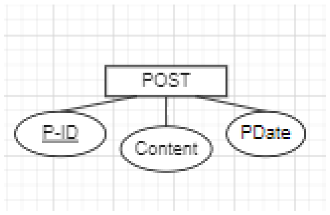
JOB_AD:



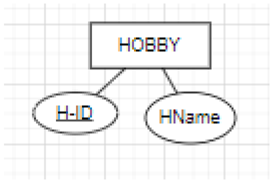
USER:



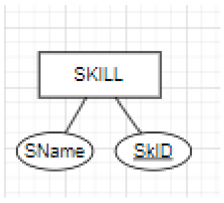
POST:



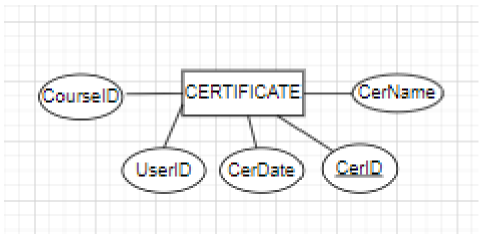
HOBBY:



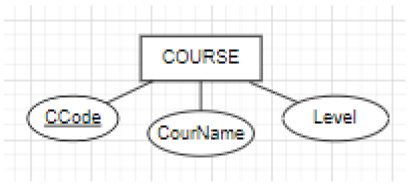
SKILL:



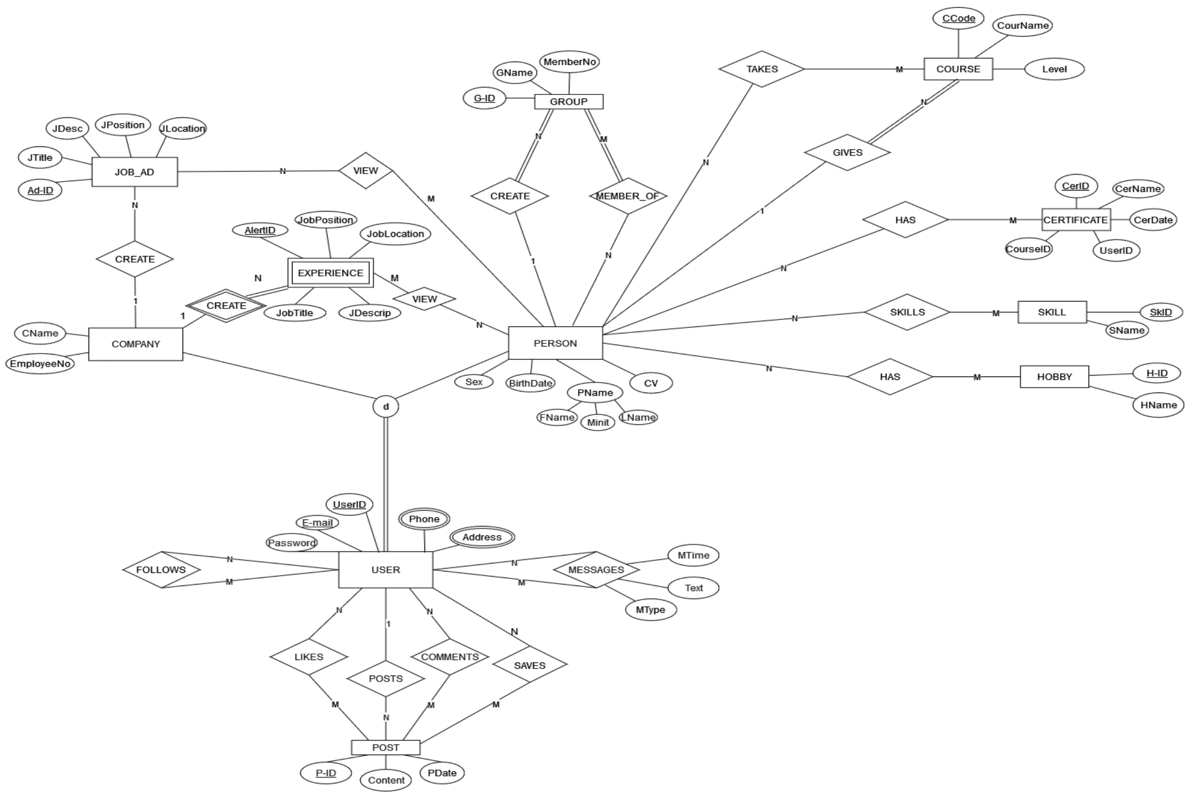
CERTIFICATE:



COURSE:



FINAL DESIGN LINKEDIN:

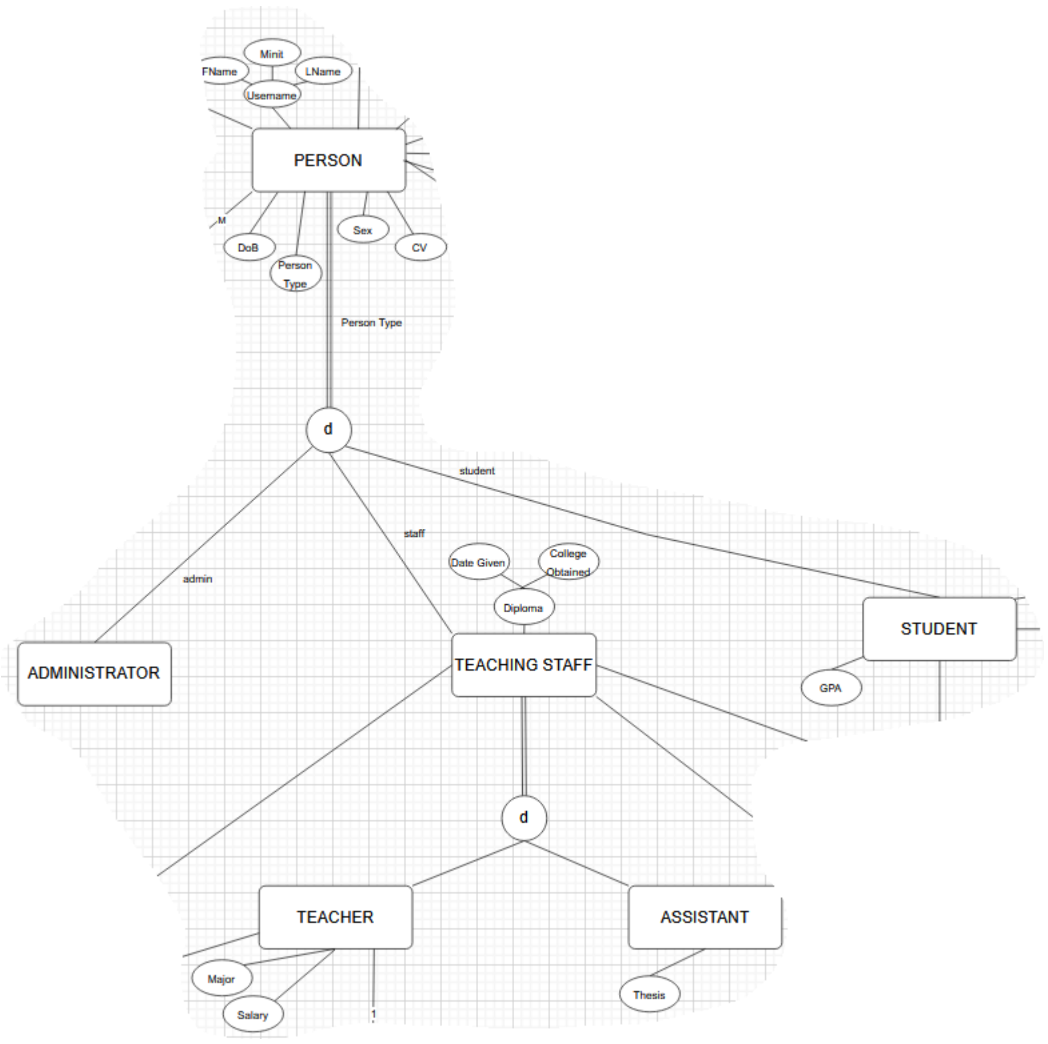


Moodle Design:

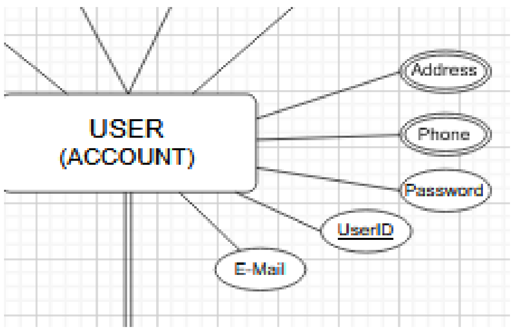
We have discussed about the data requirements of this model in the previous question.

Here is the Initial Design of our Moodle Database:

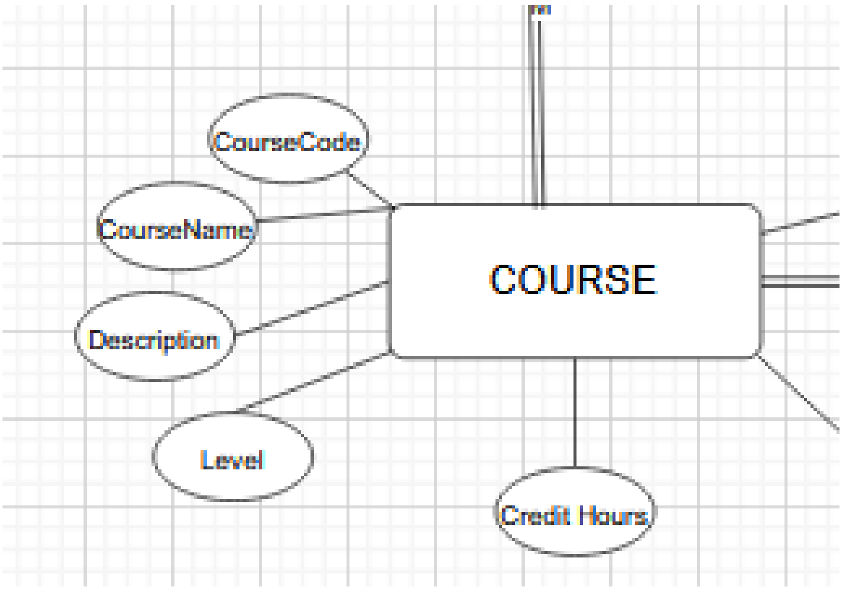
PERSON:



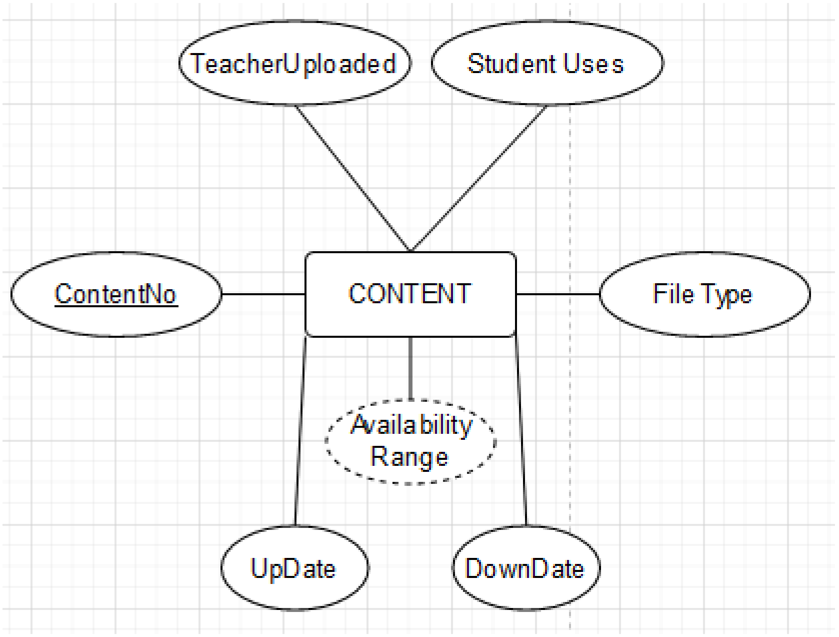
ACCOUNT:



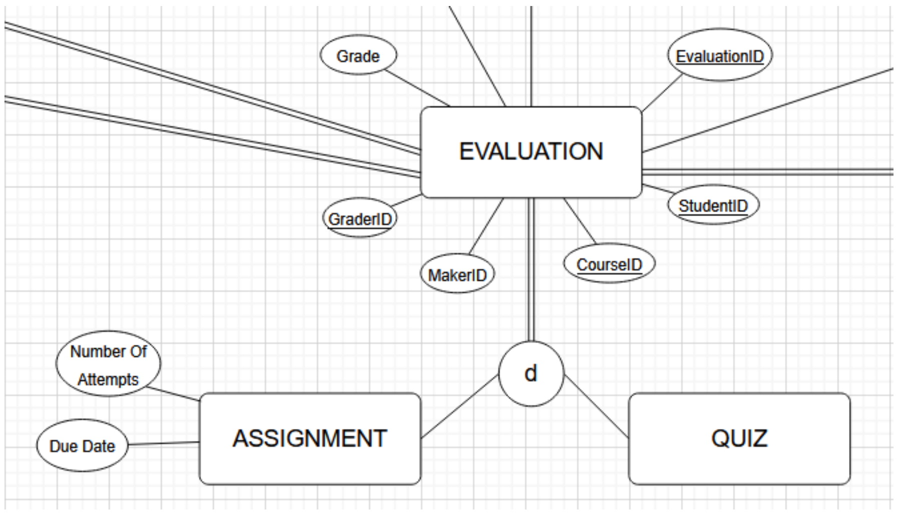
COURSE:



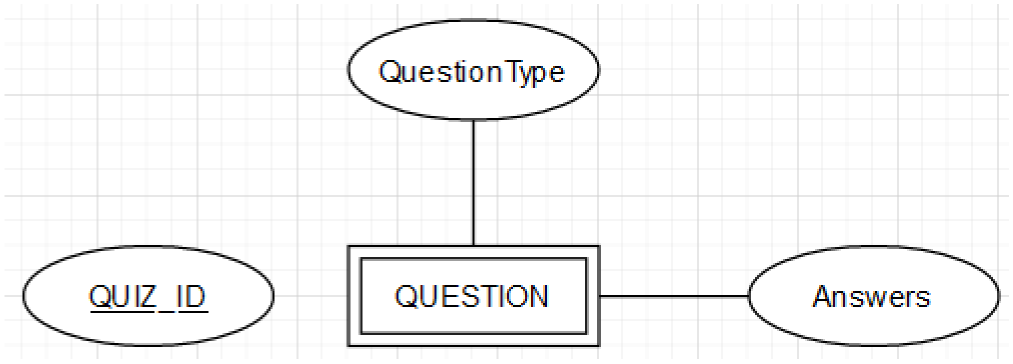
CONTENT:



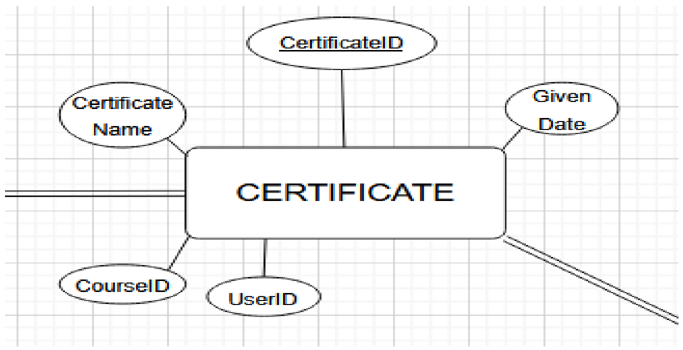
EVALUATION:



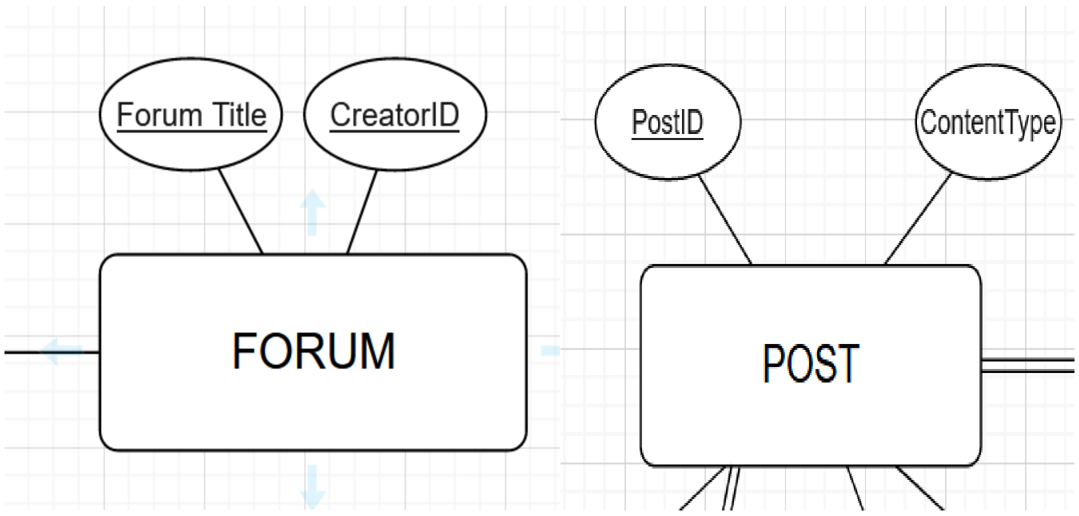
QUESTIONS:



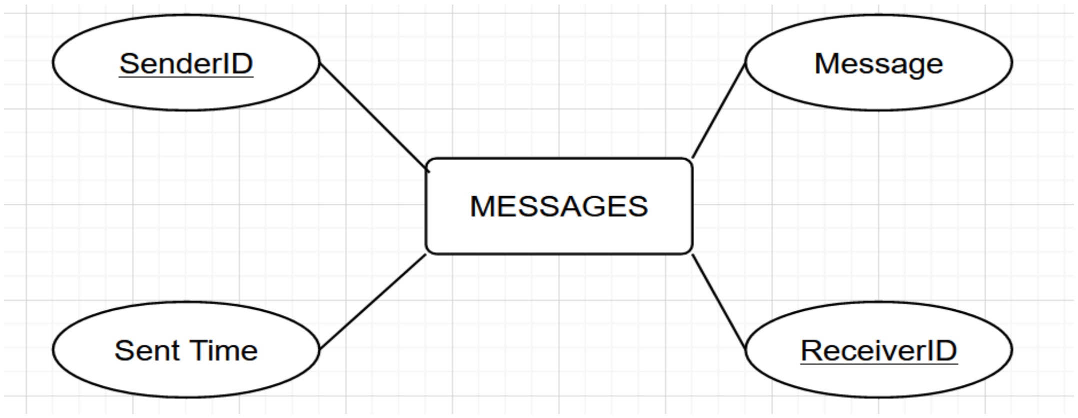
CERTIFICATE:



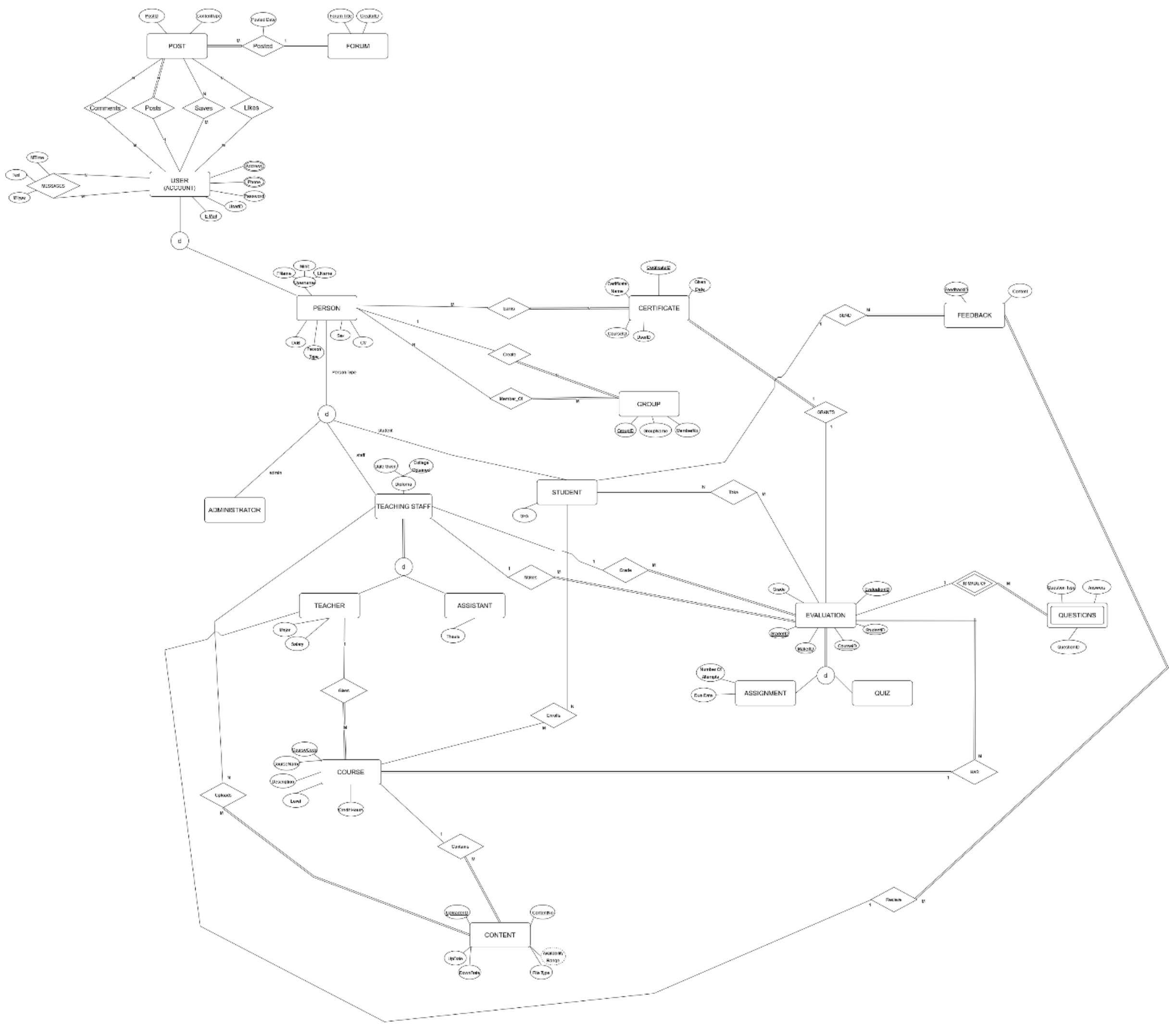
FORUM:



MESSAGES:



FINAL DESIGN MOODLE:



LinkedInMoodle:

USER(ACCOUNT):

AccountID, Password, Role, E-Mail

PERSON:

PersonID, Username(FName, Minit, LName), DOB, Address, Phone(Multival), Sex, Job, CV

```
                         PERSON

|                    |                     |

TEACHING STAFF       ADMIN                 STUDENT

|           |

Teacher     Assistant
```

MESSAGES:

MType, MTime, Text

FOLLOWS:

FTime

TEACHING STAFF:

Diploma(Date Given, College Obtained)

TEACHER:

Major, Salary, TeacherID

ADMINISTRATOR:

AdminID

ASSISTANT:

Thesis, AssistantID

STUDENT:

GPA

COMPANY:

CompanyID, CName, Address(Multival), Phone, EmployeeNumber

EXPERIENCE: (weak entity)

Start Date, End Date, Position

JOB ALERT: (weak entity)

AlertID, Job Position, Job Title, Job Description, Job Location

GROUP:

GroupID, Group Name, MemberNo (id falan gibi bir şey mi??)

FORUM:

Post ile bağlantılı, aralarıında bir relation var

Forum Title, CreatorID

POST:

PostID, PosterID, ContentType (pdate , relationship üzerinde kaydedilecek)

COURSE:

CourseCode, CourseName, TeacherID, Description, Level, CreditHours

CONTENT:

UploaderID, ContentNo, UpDate, DownDate, (Availability Range), File Type

EVALUATION:

<u>EvaluationID</u>, StudentID, GraderID, Grade

| |

Assignment   **Quiz**

ASSIGNMENT:

Number Of Attempts, Due Date

QUIZ:

CERTIFICATE:

<u>Certificate Name</u>, CourseID, UserID, GivenDate

FEEDBACK:

<u>FeedbackID</u>, Content

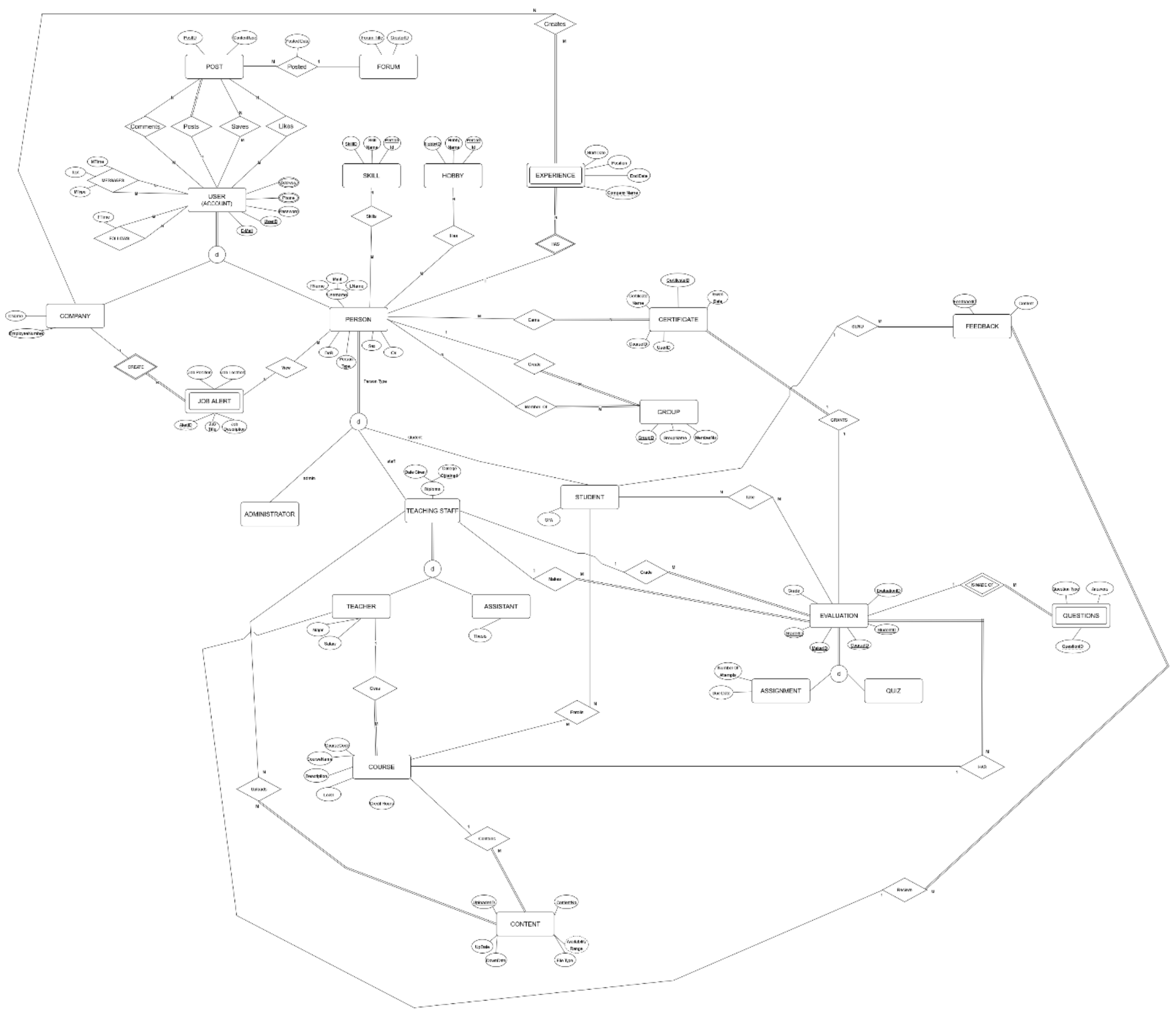
HOBBY:

<u>HobbyID</u>, Hobby Name, UserID

SKILL:

<u>SkillID</u>, Skill Name, UserID

FINAL EER DESIGN:



LinkedInMoodle Mapping:

1st Iteration

STEP 1 MAPPING OF REGULAR ENTITY TYPES:

- **FORUM** (Forum Title, CreatorID) creator id taaa user Id'ye kadar uzanan bir fk
- **POST** (Post ID, Content Type)
- **SKILL** (Skill ID, Skill Name, Person ID) person ID, person'daki PersonID'ye uzanan bir fk
- **HOBBY** (Hobby ID, Hobby Name, Person ID) aynı mantık
- **CERTIFICATE** (Certificate ID, Certificate Name, Given Date, Course ID, Person ID) Course ID ve Person ID FK

- **GROUP** (<u>Group ID</u>, Group Name, Member No)
- **COURSE** (<u>Course Code</u>, Course Name, Description, Level, CreditHours)
- **CONTENT** (<u>Content No</u>, File Type, Up Date, Down Date, Uploader ID) dervied attribute sor
- **FEEDBACK** (<u>Feedback ID</u>, Content)

STEP 2 MAPPING OF WEAK ENTITY TYPES

-

STEP 3 MAPPING OF 1:1

-

STEP 4 MAPPING OF 1:N

- Post M POSTED 1 Forum
  - **POST** (<u>Post ID</u>, Content Type, Forum Title, Posted Date) forum title, forum'dan gelen bir foreign key, post update edilmiştir.
- Course 1 Contains M Content
  - **CONTENT** (<u>Content No</u>, File Type, Up Date, Down Date, Uploader ID, Course Code), course code, course'dan gelen bir foreign key, content update edilmiştir.

STEP 5 MAPPING OF M:N

-

STEP 6 MAPPING OF Multivalues Attributes

-

STEP 7 MAPPING OF Ternary

-

STEP 8 MAPPING OF Multiple Relations and stuff

-Mandatory, disjoint (USER)

8.A ile yapacak olursak:

Superclass'ın PK'ini alıyorsun, Subclass'ın Attribute'u olarak yazıyorsun FK olacak şekilde. Subclass'ın Key attribute'u, kendisinin PK'I olacak

USER:

(<u>User ID</u>, Password, E-Mail, multival attr var unutma)

COMPANY:

(<u>Company User ID</u>, Company Name, EmployeeNumber)

PERSON:

(<u>Person User ID</u>, FName, Minit, LName, DoB, Sex, Cv, Person Type)

8B

COMPANY(<u>User ID</u>, Password, E-Mail, multival attr var unutma, CompanyName, EmployeeNumber)

PERSON(<u>User ID</u>, Password, E-Mail, multival attr var unutma, FName, Minit, LName, DoB, Sex, Cv, Person Type)

STEP 9 MAPPING OF UNION TYPES

-

2nd Iteration

STEP 1 MAPPING OF REGULAR ENTITY TYPES:

-

STEP 2 MAPPING OF WEAK ENTITY TYPES

- **JOB ALERT** (Alert ID, Company Alert User ID, Job Position, Job Location, Job Title, Job Description)
- **EXPERIENCE** (Company Name, Experience Person User ID, Position, Start Date, End Date)

STEP 3 MAPPING OF 1:1

-

STEP 4 MAPPING OF 1:N

- **POST** (Post ID, Content Type, Forum Title, User ID)
- **GROUP** (Group ID, Group Name, Member No, Person User ID)
- **CONTENT** (Content No, File Type, Up Date, Down Date, Uploader ID, CourseCode)

STEP 5 MAPPING OF M:N

- **Company M Creates N Experience**
  - **CREATES**(Company Name, PersonUserIDExperience, CompanyUserID)
- **User M Messages N User**
  - **MESSAGES**(SenderUserID, ReceiverUserID, MTime, Text, MType) //unary many to many
- **User M Follows N User**

- **FOLLOWS(**FollowerUserID, FollowedUserID, FollowTime**)**

- **Post N Comments M User**
  - **COMMENTS**(UserID, PostID)

- **Post N Saves M User**
  - **SAVES**(UserID, PostID)
- **Post N Likes M User**
  - **LIKES**(UserID, PostID)

- **Person M Skills N Skill**
  - **SKILLS**(PersonID, SkillID)

- **Person M Has N Hobby**
  - **HAS**(PersonID, HobbyID)
- **Person M Earns N Certificate**
  - **EARNS**(PersonID, CertificateID)

- **Person M MemberOf N Group**
  - **MEMBER_OF**(PersonID, GroupID

- **Person M View N Job Alert**
  - **VIEW**(PersonID, AlertID)

STEP 6 MAPPING OF Multivalues Attributes

- **USER_PHONE**(UserID, UserPhone)
- **USER_ADDRESS**(UserID, UserAddress)

STEP 7 MAPPING OF Ternary

-

STEP 8 MAPPING OF Multiple Relations and stuff

-Mandatory, disjoint (PERSON)

8A:

ADMINISTRATOR(UserID)

TEACHING_STAFF(UserID, DateGiven, CollegeObtained)

STUDENT(UserID, GPA)

STEP 9 MAPPING OF UNION TYPES

-

3rd Iteration

STEP 1 MAPPING OF REGULAR ENTITY TYPES:

-

STEP 2 MAPPING OF WEAK ENTITY TYPES

-

STEP 3 MAPPING OF 1:1

-

STEP 4 MAPPING OF 1:N

- **FEEDBACK** (<u>Feedback ID</u>, Content, StudentUserID)

STEP 5 MAPPING OF M:N

- **Teaching Staff M Uploads N Content**
    - **Uploads**(<u>TeachStaffID, ContentNo</u>)

STEP 6 MAPPING OF Multivalues Attributes

-

STEP 7 MAPPING OF Ternary

-

STEP 8 MAPPING OF Multiple Relations and stuff

8A:

TEACHING_STAFF(<u>UserID</u>, DiplomaDateGiven, CollegeObtained, TeachStaffType)

TEACHER(<u>UserID</u>, Major, Salary)

ASSISTANT(<u>UserID</u>, Thesis)

8A:

EVALUATION(<u>StudentUserID(studentID), EvaluationID,</u> CourseID, GraderID, Grade, MakerID)

ASSIGNMENT(<u>UserID,EvaluationID</u>, DueDate, NumberofAttempts)

QUIZ(<u>UserID, EvaluationID</u>)

---- 8b version

8B

TEACHER(UserID, DateGiven, CollegeObtained,Major, Salary)

ASSISTANT(UserID, DateGiven, CollegeObtained,Thesis)

8B

ASSIGNMENT(UserID,EvaluationID, StudentID, CourseID, GraderID, Grade, DueDate, NumberofAttempts)

QUIZ(UserID, EvaluationID StudentID, CourseID, GraderID, Grade)

STEP 9 MAPPING OF UNION TYPES

-

4th Iteration

STEP 1 MAPPING OF REGULAR ENTITY TYPES:

-

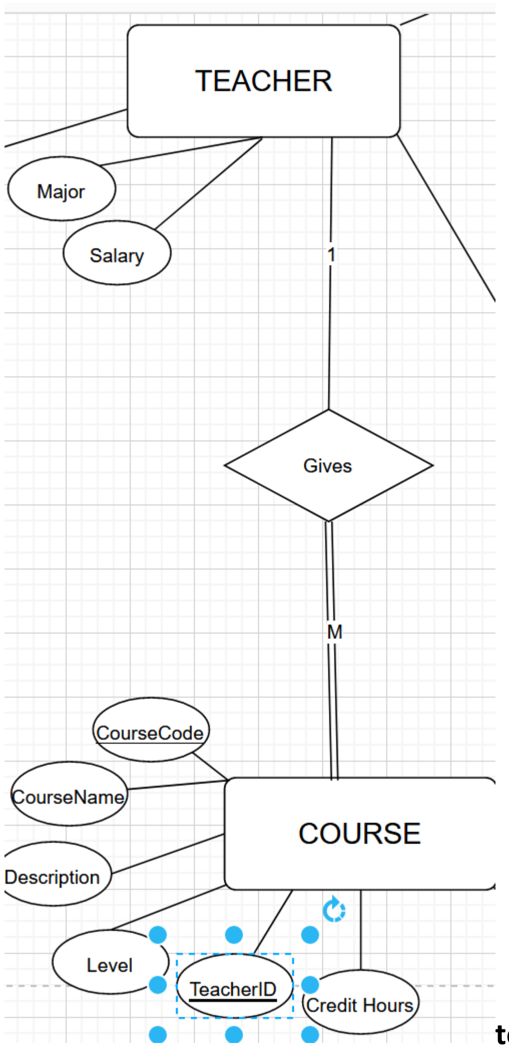
STEP 2 MAPPING OF WEAK ENTITY TYPES

QUESTIONS(QuestionID, ExamEvaluationID, QuestionType, Answers)

STEP 3 MAPPING OF 1:1

- **Evaluation 1 Grants 1 Certificate**
  - **CERTIFICATE** (Certificate ID, Certificate Name, Given Date, Course ID, Person ID, StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), GraderID(teachingStaffID))

STEP 4 MAPPING OF 1:N

- **Teacher 1 Gives M Course**
  - COURSE (<u>Course Code,</u> Course Name, Description, Level, CreditHours, TeacherUserID)
  - 

teacherID EER MODELDEN kaldırıldı, veri tekrarından dolayı ötürü.
- **TeachingStaff 1 Makes N Evaluation**
  - **EVALUATION**(<u>StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID,</u> Grade)
- **TeachingStaff 1 Grade M Evaluation**
  - **EVALUATION**(<u>StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID, GraderID(teachingStaffID),</u> Grade)

Course 1 Has M Evaluation

- **EVALUATION**(<u>StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID(courseCode), GraderID(teachingStaffID),</u> Grade)

- **Teacher 1 Receive M FeedBack**
  - **FEEDBACK** (Feedback ID, Content, StudentUserID, TeacherUserID) //RECEIVE

STEP 5 MAPPING OF M:N

- **Course M Enrolls N Student**
  - **Enrolls**(CourseCode, StudenUsertID)
- **Student M Take N Evaluation**
  - **Take**(StudentUserID, StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID(courseCode), GraderID(teachingStaffID))

STEP 6 MAPPING OF Multivalues Attributes

-

STEP 7 MAPPING OF Ternary

-

STEP 8 MAPPING OF Multiple Relations and stuff

-

STEP 9 MAPPING OF UNION TYPES

-

LinkedInMoodle Relation Schema:

- **//FORUM** (Forum Title, CreatorID)
- **//POST** (Post ID, Content Type, Forum Title, User ID)
- **//SKILL** (Skill ID, Skill Name, Person ID)
- **//HOBBY** (Hobby ID, Hobby Name, Person ID)
- **//GROUP** (Group ID, Group Name, Member No, Person User ID)
- **//COURSE** (Course Code, Course Name, Description, Level, CreditHours, TeacherUserID)
- **//CONTENT** (Content No, File Type, Up Date, Down Date, Uploader ID) dervied attribute sor
- **//FEEDBACK** (Feedback ID, Content, StudentUserID, TeacherUserID)
- **//QUESTIONS**(QuestionID, ExamEvaluationID, QuestionType, Answers)
- **//JOB ALERT (**Alert ID, Company Alert User ID, Job Position, Job Location, Job Title, Job Description**)**
- **//EXPERIENCE (**Company Name, Experience Person User ID, Position, Start Date, End Date**)**
- **//CERTIFICATE** (Certificate ID, Certificate Name, Given Date, Person ID, StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID(courseCode), GraderID(teachingStaffID))
- **//EVALUATION**(StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID(courseCode), GraderID(teachingStaffID), Grade)
- **//ENROLLS**(CourseCode, StudenUsertID)
- **//TAKE**(StudentUserID(studentUserID), EvaluationID, MakerID(teachingStaffID), CourseID(courseCode), GraderID(teachingStaffID))
- **//ASSIGNMENT**(UserID,EvaluationID, DueDate, NumberofAttempts)
- **//QUIZ**(UserID, EvaluationID)
- **//TEACHING_STAFF**(UserID, DiplomaDateGiven, CollegeObtained, TeachStaffType)
- **//ADMINISTRATOR(**AdminID**)**
- **//TEACHER**(UserID, Major, Salary)
- **//ASSISTANT**(UserID, Thesis)
- **//STUDENT**(UserID, GPA)
- **//UPLOADS**(TeachStaffID, ContentNo)

- **//USER_PHONE(**UserID, UserPhone **)**
- **//USER_ADDRESS(**UserID, UserAddress **)**
- **//VIEW(**PersonID, AlertID**)**
- **//MEMBER_OF(**PersonID, GroupID**)**
- **//EARNS**(PersonID, CertificateID)
- **//HAS(**PersonID, HobbyID**)**
- **//SKILLS**(PersonID, SkillID)
- **//LIKES**(UserID, PostID)
- **//SAVES(**UserID, PostID**)**
- **//COMMENTS**(UserID, PostID)
- **//USER**(User ID, Password, E-Mail, multival attr var unutma)
- **//COMPANY**(Company User ID, Company Name, EmployeeNumber)
- **//PERSON**(Person User ID, FName, Minit, LName, DoB, Sex, Cv, Person Type)
- **MESSAGES(**SenderUserID, RecieverUserID, MTime, Text, MType**)**

SQL DDL STATEMENTS:

```
DROP DATABASE IF EXISTS LinkedinMoodle;

CREATE DATABASE LinkedinMoodle;



#Creating LinkedinMoodle Schema

Use LinkedinMoodle;



DROP TABLE IF EXISTS USERS;

CREATE TABLE USERS(

        UserID int not null,

  Passwordd varchar(25) not null,

  EMail varchar(50) not null,

  CONSTRAINT pk_USERS primary key (UserID),

  CONSTRAINT uq_EMail UNIQUE (EMail)

);



INSERT INTO USERS VALUES ("0", "sebastian123", "sebastianvettel@gmail.com");
INSERT INTO USERS VALUES ("1", "12ferrari34", "scuderiaferrari@gmail.com");

INSERT INTO USERS VALUES ("2", "password", "gokberkayhan@gmail.com");

INSERT INTO USERS VALUES ("3", "123123", "cankan23@hotmail.com");

INSERT INTO USERS VALUES ("4", "98765", "redbullracing@gmail.com");

INSERT INTO USERS VALUES ("5", "selin35", "selinpaksoy@gmail.com");

INSERT INTO USERS VALUES ("6", "ichLaubeDatabeisieren", "dataismylife@ege.edu.tr");

INSERT INTO USERS VALUES ("7", "sevgi<3", "bestekaysi@gmail.com");

INSERT INTO USERS VALUES ("8", "intel8086", "micropro@gmail.com");

INSERT INTO USERS VALUES ("9", "stackQueue", "datastructures@gmail.com");

INSERT INTO USERS VALUES ("10", "mrDatabase", "mrdatabase@ege.edu.tr");

INSERT INTO USERS VALUES ("11", "sifresifre", "eposta@gmail.com");

INSERT INTO USERS VALUES ("12", "isletimsis", "isletimsis@yahoo.com");

INSERT INTO USERS VALUES ("13", "bilgag", "internationallove@gmail.com");

INSERT INTO USERS VALUES ("14", "nesnelbilgiler35", "oopismylife@gmail.com");


DROP TABLE IF EXISTS USER_PHONE;

CREATE TABLE USER_PHONE(

            PhoneUsersID int not null,

   PhoneNumber varchar(11) not null,

   CONSTRAINT pk_USER_PHONE primary key (PhoneUsersID, PhoneNumber),

   CONSTRAINT pf_USER_PHONE_USERS foreign key (PhoneUsersID) references USERS(UserID)

);


INSERT INTO USER_PHONE VALUES ("5", "05364418586");

INSERT INTO USER_PHONE VALUES ("5", "05325320532");


DROP TABLE IF EXISTS USER_ADDRESS;

CREATE TABLE USER_ADDRESS(

            AddressUsersID int not null,

   Address varchar(50) not null,

   CONSTRAINT pk_USER_ADDRESS primary key (AddressUsersID, Address),
   CONSTRAINT pf_USER_ADDRESS_USERS foreign key (AddressUsersID) references USERS(UserID)

);


INSERT INTO USER_ADDRESS VALUES ("1", "Via Paolo Ferrari, 85, 41121 Modena, Italy");


DROP TABLE IF EXISTS PERSON;

CREATE TABLE PERSON(

            PersonUserID int not null,

   FName varchar(17) not null,

   Minit varchar(1),

   LName varchar(17) not null,

   DoB date,

   Sex char,

   CV text,

   PersonType varchar(15) not null,

   CONSTRAINT pk_PERSON primary key (PersonUserID),

   CONSTRAINT fk_PERSON_USERS foreign key (PersonUserID) references USERS(UserID)

);


INSERT INTO PERSON VALUES ("0", "Sebastian", null, "Vettel", "1987-7-3", "M", "Toro Rosso, Red Bull Racing,
Scuderia Ferrari, Aston Martin", "Student");

INSERT INTO PERSON VALUES ("2", "Gokberk", "M", "Ayhan", "2000-11-15", "M", null, "Student");

INSERT INTO PERSON VALUES ("5", "Selin", "Z", "Paksoy", "2001-03-30", "F", "Ege uni", "Student");

INSERT INTO PERSON VALUES ("10", "William", "M", "Database", "1969-01-30", "M", "Facebook",
"Administrator");

INSERT INTO PERSON VALUES ("6", "Osman", "K", "Unalir", "1971-10-08", "M", "Ege Univ", "Teaching Staff");

INSERT INTO PERSON VALUES ("7", "Beste", "Z", "Kaysi", "1992-03-01", "F", "Dokuz Eylul", "Teaching Staff");

INSERT INTO PERSON VALUES ("8", "Sebnem", null, "Bora", "1980-04-10", "F", "Intel", "Teaching Staff");

INSERT INTO PERSON VALUES ("9", "Aybars", null, "Ugur", "1967-03-11", "M", "Izmir BSB", "Teaching Staff");

INSERT INTO PERSON VALUES ("11", "Esra", "K", "Duman", "2000-01-29", "F", null, "Student");

INSERT INTO PERSON VALUES ("12", "Aylin", "K", "Kantarci", "1980-01-30", "F", "Microsoft", "Teaching Staff");

INSERT INTO PERSON VALUES ("13", "Levent", "T", "Toker", "1960-07-15", "M", "Turk Telekom", "Teaching
Staff");
```

```
INSERT INTO PERSON VALUES ("14", "Riza", "C", "Erdur", "1963-12-31", "M", "Automata", "Teaching Staff");


DROP TABLE IF EXISTS MESSAGES;
CREATE TABLE MESSAGES(
	SenderUserID int not null,
   RecieverUserID int not null,
   MTime double not null,                    # 10.31
   MText text not null,
   MType char not null,                      # T for Text, I for Image, V for Video
   CONSTRAINT pk_MESSAGES primary key (SenderUserID, RecieverUserID),
   CONSTRAINT fk_MESSAGES_USERS1 foreign key (SenderUserID) references USERS(UserID),
   CONSTRAINT fk_MESSAGES_USERS2 foreign key (RecieverUserID) references USERS(UserID)
);

INSERT INTO MESSAGES VALUES("5","2", "18.06", "Gokberk projeyi bitirdin mi?", "T");
INSERT INTO MESSAGES VALUES("2","5", "20.03", "HAYIR.:D", "T");

DROP TABLE IF EXISTS FOLLOWS;
CREATE TABLE FOLLOWS(
	FollowerUserID int not null,
   FollowedUserID int not null,
   FollowTime decimal not null,
   CONSTRAINT pk_FOLLOWS primary key (FollowerUserID, FollowedUserID),
   CONSTRAINT fk_FOLLOWS_USERS1 foreign key (FollowerUserID) references USERS(UserID),
   CONSTRAINT fk_FOLLOWS_USERS2 foreign key (FollowedUserID) references USERS(UserID)
);

DROP TABLE IF EXISTS GRUP;
CREATE TABLE GRUP(
	GroupID int not null,
   GroupName varchar(25) not null,

   MemberNo int not null,

   CreatorPersonUserID int not null,

   CONSTRAINT pk_GRUP primary key (GroupID),

   CONSTRAINT fk_GRUP_PERSON foreign key (CreatorPersonUserID) references PERSON(PersonUserID)

);


INSERT INTO GRUP VALUES ("3", "TIFOSI IZMIR", "1", "2");

INSERT INTO GRUP VALUES ("1", "DBProje Grubu", "1", "5");


DROP TABLE IF EXISTS MEMBER_OF;

CREATE TABLE MEMBER_OF(

	MemberPersonUserID int not null,

   MemberGroupID int not null,

   CONSTRAINT pk_MEMBER_OF primary key (MemberPersonUserID, MemberGroupID),

   CONSTRAINT fk_MEMBER_OF_PERSON foreign key (MemberPersonUserID) references
PERSON(PersonUserID),

   CONSTRAINT fk_MEMBER_OF_GRUP foreign key (MemberGroupID) references GRUP(GroupID)

);


CREATE TRIGGER after_member_attend

AFTER INSERT ON MEMBER_OF

FOR EACH ROW

	UPDATE GRUP,MEMBER_OF SET MemberNo = MemberNo + 1

   WHERE  MemberNo = GRUP.MemberNo AND MEMBER_OF.MemberGroupID = GRUP.GroupID;


INSERT INTO MEMBER_OF VALUES ("0", "3");

INSERT INTO MEMBER_OF VALUES ("5", "3");

INSERT INTO MEMBER_OF VALUES ("10", "3");			# tifosi izmir 4 kisi olmali

INSERT INTO MEMBER_OF VALUES ("0", "1");

INSERT INTO MEMBER_OF VALUES ("7", "1");			# db proje grup 3 kisi olmali

INSERT INTO MEMBER_OF VALUES ("8", "1");
INSERT INTO MEMBER_OF VALUES ("9", "1");


DROP TABLE IF EXISTS COMPANY;

CREATE TABLE COMPANY(

	CompanyUserID int not null,

   CompanyName varchar(40),

   EmployeeNumber int,

   CONSTRAINT uq_CompanyName UNIQUE (CompanyName),

   CONSTRAINT pk_COMPANY primary key (CompanyUserID),

   CONSTRAINT fk_COMPANY_USERS foreign key (CompanyUserID) references USERS(UserID)

);


INSERT INTO COMPANY VALUES ("1", "Scuderia Ferrari", "250");

INSERT INTO COMPANY VALUES ("4", "Red Bull Racing Honda", "600");


DROP TABLE IF EXISTS EXPERIENCE;

CREATE TABLE EXPERIENCE(

	Company int not null,				# reference to company id

   ExperienceUserID int not null,      # reference to PersonUserID

   WorkingPosition varchar(30) not null,

   StartDate date not null,

   EndDate date,

   CONSTRAINT pk_EXPERIENCE primary key (Company, ExperienceUserID),

   CONSTRAINT fk_EXPERIENCE_COMPANY foreign key (Company) references COMPANY(CompanyUserID),

   CONSTRAINT fk_EXPERIENCE_PERSON foreign key (ExperienceUserID) references PERSON(PersonUserID)

);


CREATE TRIGGER after_insert_experience

AFTER INSERT ON EXPERIENCE

FOR EACH ROW

	UPDATE COMPANY,EXPERIENCE SET EmployeeNumber = EmployeeNumber + 1 WHERE
EmployeeNumber = COMPANY.EmployeeNumber
```

```
                    AND EXPERIENCE.Company = COMPANY.CompanyUserID;


INSERT INTO EXPERIENCE VALUES("4", "2", "internship", "2020-07-05", "2021-01-31");

INSERT INTO EXPERIENCE VALUES("4", "0", "longterm", "2018-08-11", " 2022-08-15");

INSERT INTO EXPERIENCE VALUES("1", "5", "internship", "2022-01-01", null);


DROP TABLE IF EXISTS JOB_ALERT;

CREATE TABLE JOB_ALERT(

        AlertID int not null,

   AlertingCompanyUserID int not null,

   JobPosition varchar(20) not null,

   JobLocation varchar(15) not null,

   JobTitle varchar(15) not null,

   JobDescription text,

   CONSTRAINT pk_JOB_ALERT primary key (AlertID, AlertingCompanyUserID),

   CONSTRAINT fk_JOB_ALERT_COMPANY foreign key (AlertingCompanyUserID) references
COMPANY(CompanyUserID)

);


INSERT INTO JOB_ALERT VALUES ("1000", "4", "Tyre Expert", "London", "Junior", "Can speak english");

INSERT INTO JOB_ALERT VALUES ("999", "1", "Data Engineer", "Maranello", "Senior", "Grazie regazzi");

INSERT INTO JOB_ALERT VALUES ("998", "4", "Pilot", "London", "Junior", "Drivers licence req.");

INSERT INTO JOB_ALERT VALUES ("997", "1", "Pilot", "London", "Senior", "No requirement needed");

INSERT INTO JOB_ALERT VALUES ("996", "1", "Team Coach", "Maranello", "New Grad", "Speaks Italian");


DROP TABLE IF EXISTS VIEW_ALERT;

CREATE TABLE VIEW_ALERT(

        ViewingPersonID int not null,

        ViewedAlertID int not null,

   CONSTRAINT pk_VIEW_ALERT primary key (ViewingPersonID, ViewedAlertID),

   CONSTRAINT fk_VIEW_ALERT_PERSON foreign key (ViewingPersonID) references PERSON(PersonUserID),
   CONSTRAINT fk_VIEW_ALERT_JOB_ALERT foreign key (ViewedAlertID) references JOB_ALERT(AlertID) ON
DELETE CASCADE

);

INSERT INTO VIEW_ALERT VALUES("2", "999");


DROP TABLE IF EXISTS FORUM;

CREATE TABLE FORUM(

            ForumTitle varchar(20) not null,

   CreatorUserID int not null,

   CONSTRAINT pk_FORUM primary key (ForumTitle),

   CONSTRAINT fk_FORUM_USERS foreign key (CreatorUserID) references USERS(UserID)

);


INSERT INTO FORUM VALUES ("Database Management", "2");


DROP TABLE IF EXISTS POST;

CREATE TABLE POST(

            PostID int not null,

   ContentType char,                                    # V for Video, I for Image, T for Text

   ForumTitle varchar(20) not null,

   PosterUserID int not null,

   CONSTRAINT pk_POST primary key (PostID),

   CONSTRAINT fk_POST_FORUM foreign key (ForumTitle) references FORUM(ForumTitle),

   CONSTRAINT fk_POST_USERS foreign key (PosterUserID) references USERS(UserID)

);


INSERT INTO POST VALUES ("11", "I", "Database Management", "2");


DROP TABLE IF EXISTS COMMENTS;

CREATE TABLE COMMENTS(

            CommentingUserID int not null,

   CommentedPostID int not null,
   CONSTRAINT pk_COMMENTS primary key (CommentingUserID, CommentedPostID),
   CONSTRAINT fk_COMMENTS_USERS foreign key (CommentingUserID) references USERS(UserID)
);

INSERT INTO COMMENTS VALUES ("5", "41");

DROP TABLE IF EXISTS SAVES;
CREATE TABLE SAVES(
            SavingUserID int not null,
   SavedPostID int not null,
   CONSTRAINT pk_SAVES primary key (SavingUserID, SavedPostID),
   CONSTRAINT fk_SAVES_USERS foreign key (SavingUserID) references USERS(UserID)
);

INSERT INTO SAVES VALUES ("0", "11");

DROP TABLE IF EXISTS LIKES;
CREATE TABLE LIKES(
            LikingUserID int not null,
   LikedPostId int not null,
   CONSTRAINT pk_LIKES primary key (LikingUserID, LikedPostID),
   CONSTRAINT fk_LIKES_USERS foreign key (LikingUserID) references USERS(UserID)
);

INSERT INTO LIKES VALUES ("5", "17");

DROP TABLE IF EXISTS SKILL;
CREATE TABLE SKILL(
            SkillID varchar(5) not null,
   SkillName varchar(15) not null,
   SkillPersonUserID int not null,
   CONSTRAINT pk_SKILL primary key (SkillID),

   CONSTRAINT fk_SKILL_USERS foreign key (SkillPersonUserID) references PERSON(PersonUserID) ON DELETE
CASCADE

);


INSERT INTO SKILL VALUES ("cdg", "Coding", "2");

INSERT INTO SKILL VALUES ("sgn", "Singing", "5");


DROP TABLE IF EXISTS HOBBY;

CREATE TABLE HOBBY(

        HobbyID varchar(5) not null,                    # cdg (coding) falan filan

   HobbyName varchar(15) not null,

   HobbyPersonUserID int not null,

   CONSTRAINT pk_HOBBY primary key (HobbyID),

   CONSTRAINT fk_Hobby_USERS foreign key (HobbyPersonUserID) references PERSON(PersonUserID) ON
DELETE CASCADE

);


INSERT INTO HOBBY VALUES ("drvg", "Driving", "0");

INSERT INTO HOBBY VALUES ("read", "Reading", "8");

INSERT INTO HOBBY VALUES ("trvl", "Traveling", "9");


DROP TABLE IF EXISTS HAS;

CREATE TABLE HAS(

        HobbysPersonUserID int not null,

   HHobbyID varchar(5) not null,                # cdg (coding) falan filan

   CONSTRAINT pk_HAS primary key (HobbysPersonUserID, HHobbyID),

   CONSTRAINT fk_HAS_PERSON foreign key (HobbysPersonUserID) references PERSON(PersonUserID),

   CONSTRAINT fk_HAS_HOBBY foreign key (HHobbyID) references HOBBY(HobbyID)

);


INSERT INTO HAS VALUES ("0", "drvg");
INSERT INTO HAS VALUES ("8", "read");

INSERT INTO HAS VALUES ("9", "trvl");


DROP TABLE IF EXISTS ADMINISTRATOR;

CREATE TABLE ADMINISTRATOR(

            AdminID int not null,

   CONSTRAINT pk_ADMINISTRATOR primary key (AdminID),

   CONSTRAINT fk_ADMINISTRATOR_USERS foreign key (AdminID) references PERSON(PersonUserID)

);


INSERT INTO ADMINISTRATOR VALUES ("10");


DROP TABLE IF EXISTS TEACHING_STAFF;

CREATE TABLE TEACHING_STAFF(

            TeachingUserID int not null,

   DiplomaGivenDate date not null,

   CollegeObtained varchar(25) not null,

   TeachingStaffType char not null,                              # T for teacher, A for assistant

   CONSTRAINT pk_TEACHING_STAFF primary key (TeachingUserID),

   CONSTRAINT fk_TEACHING_STAFF_USER foreign key (TeachingUserID) references PERSON(PersonUserID)

);


INSERT INTO TEACHING_STAFF VALUES ("6", "1993-06-06", "Ege Univ", "T");

INSERT INTO TEACHING_STAFF VALUES ("8", "1996-10-01", "Dokuz Eylul", "T");

INSERT INTO TEACHING_STAFF VALUES ("7", "2010-06-07", "ITU", "A");

INSERT INTO TEACHING_STAFF VALUES ("9", "1996-01-10", "Ege Univ", "T");

INSERT INTO TEACHING_STAFF VALUES ("12", "1993-05-11", "Dokuz Eylul", "T");

INSERT INTO TEACHING_STAFF VALUES ("13", "1989-01-21", "Ege Univ", "T");

INSERT INTO TEACHING_STAFF VALUES ("14", "1991-01-31", "Ege Univ", "T");


DROP TABLE IF EXISTS STUDENT;
```

```
CREATE TABLE STUDENT(
            StudentUserID int not null,
   GPA double,
   CONSTRAINT pk_STUDENT primary key (StudentUserID),
   CONSTRAINT fk_STUDENT_USERS foreign key (StudentUserID) references Person(PersonUserID)
);

INSERT INTO STUDENT VALUES ("5", "3.31");
INSERT INTO STUDENT VALUES ("2", "3.62");
INSERT INTO STUDENT VALUES ("0", "2.93");
INSERT INTO STUDENT VALUES ("11", "3.01");

DROP TABLE IF EXISTS TEACHER;
CREATE TABLE TEACHER(
            TeacherUserID int not null,
   Major varchar(20) not null,
   Salary double not null CHECK (Salary >= 0.0),
   CONSTRAINT pk_TEACHER primary key (TeacherUserID),
   CONSTRAINT fk_TEACHER_USER foreign key (TeacherUserID) references TEACHING_STAFF(TeachingUserID)
);

INSERT INTO TEACHER VALUES ("6", "Database Management", "30000.00");
INSERT INTO TEACHER VALUES ("8", "Microprocessors", "25000.00");
INSERT INTO TEACHER VALUES ("9", "Data Structures", "27000.00");
INSERT INTO TEACHER VALUES ("12", "Operating Systems", "28000.00");
INSERT INTO TEACHER VALUES ("13", "Computer Networks", "51000.00");
INSERT INTO TEACHER VALUES ("14", "OOP", "31000.00");

DROP TABLE IF EXISTS FEEDBACK;
CREATE TABLE FEEDBACK(
            FeedbackID int not null,
   Content text not null,

   SendingStudentUserID int not null,

   RecievingTeacherUserID int not null,

   CONSTRAINT pk_FEEDBACK primary key (FeedbackID),

   CONSTRAINT fk_FEEDBACK_STUDENT foreign key (SendingStudentUserID) references
STUDENT(StudentUserID),

   CONSTRAINT fk_FEEDBACK_TEACHER foreign key (RecievingTeacherUserID) references
TEACHER(TeacherUserID)

);

INSERT INTO FEEDBACK VALUES ("15", "Soru eksik verilmis!", "5", "8");


DROP TABLE IF EXISTS ASSISTANT;

CREATE TABLE ASSISTANT(

            AssistantUserID int not null,

   Thesis text not null,

   CONSTRAINT pk_ASSISTANT primary key (AssistantUserID),

   CONSTRAINT fk_ASSISTANT_USERS foreign key (AssistantUserID) references
TEACHING_STAFF(TeachingUserID)

);


INSERT INTO ASSISTANT VALUES ("7", "Cizge Teorisi");


DROP TABLE IF EXISTS COURSE;

CREATE TABLE COURSE(

            CourseCode int not null,

   CourseName varchar(25) not null,

   CourseDescription text,

   CourseLevel double not null,

   CreditHours int not null CHECK (CreditHours >= 1),

   CourseTeacherUserID int not null,

   StudentAmount int not null,

   CONSTRAINT pk_COURSE primary key (CourseCode),
   CONSTRAINT fk_COURSE foreign key (CourseTeacherUserID) references TEACHER(TeacherUserID) ON DELETE
CASCADE

);



INSERT INTO COURSE VALUES ("420", "Microprocessors", "Welcome to MicroProcessor class", "3.1", "5", "8",
"0");

INSERT INTO COURSE VALUES ("538", "Data Structures", "Theory of data structures", "2.1", "6", "9", "0");

INSERT INTO COURSE VALUES ("111", "Operating Systems", "History of operating systems and their usings",
"3.1", "4", "12", "0");

INSERT INTO COURSE VALUES ("121", "Computer Networks", "History of operating systems and their usings",
"3.1", "4", "13", "0");

INSERT INTO COURSE VALUES ("131", "OOP", "History of operating systems and their usings", "3.1", "5", "14",
"0");



DROP TABLE IF EXISTS CONTENT;

CREATE TABLE CONTENT(

        ContentNo int not null,

   FileType varchar(5) not null,

   UploadDate date not null,

   DownDate date ,

   UploaderUserID int not null,

   CONSTRAINT pk_CONTENT primary key (ContentNo),

   CONSTRAINT fk_CONTENT_TEACHING_STAFF foreign key (UploaderUserID) references
TEACHING_STAFF(TeachingUserID)

);

INSERT INTO CONTENT VALUES ("13", "pdf", "2022-02-02", null, "8");

INSERT INTO CONTENT VALUES ("11", "jpg", "2022-01-28", "2022-02-01", "9");

INSERT INTO CONTENT VALUES ("911", "pdf", "2022-02-02", null, "7");




DROP TABLE IF EXISTS UPLOADS;

CREATE TABLE UPLOADS(

        UppingTeachingStaffUserID int not null,

   UppedContentNo int not null,
   CONSTRAINT pk_UPLOADS primary key (UppingTeachingStaffUserID),
   CONSTRAINT fk_UPLOADS_TEACHING_STAFF foreign key (UppedContentNo) references
TEACHING_STAFF(TeachingUserID)
);
INSERT INTO UPLOADS VALUES ("6", "7");
INSERT INTO UPLOADS VALUES ("9", "6");


DROP TABLE IF EXISTS EVALUATION;
CREATE TABLE EVALUATION(
            EvaluationID int not null,
   TakingStudentUserID int not null,                    # student id
   MakerUserID int not null,                                  # teaching staff id who makes this
evaluation
   CourseID int not null,                                     # course code
   GraderUserID int not null,                                          # teaching staff id who grades
this evaluation
   Grade int not null CHECK (Grade < 101),
   CONSTRAINT pk_EVALUATION primary key (EvaluationID, TakingStudentUserID, MakerUserID, CourseID,
GraderUserID),
   CONSTRAINT fk_EVALUATION_TEACHING_STAFF1 foreign key (MakerUserID) references
TEACHING_STAFF(TeachingUserID),
            CONSTRAINT fk_EVALUATION_TEACHING_STAFF2 foreign key (GraderUserID) references
TEACHING_STAFF(TeachingUserID),
            CONSTRAINT fk_EVALUATION_COURSE foreign key (CourseID) references COURSE(CourseCode),
   CONSTRAINT fk_EVALUATION_STUDENT foreign key (TakingStudentUserID) references
STUDENT(StudentUserID)
);


INSERT INTO EVALUATION VALUES ("1", "0", "6", "420", "6", "22");
INSERT INTO EVALUATION VALUES ("1", "11", "6", "420", "6", "59");
INSERT INTO EVALUATION VALUES ("1", "2", "6", "420", "8", "47");
INSERT INTO EVALUATION VALUES ("1", "5", "6", "420", "8", "95");
INSERT INTO EVALUATION VALUES ("6", "5", "12", "111", "12", "41");
INSERT INTO EVALUATION VALUES ("3", "5", "13", "121", "13", "60");
INSERT INTO EVALUATION VALUES ("4", "5", "14", "131", "14", "90");
INSERT INTO EVALUATION VALUES ("2", "2", "9", "538", "9", "100");


DROP TABLE IF EXISTS ASSIGNMENT;

CREATE TABLE ASSIGNMENT(

        ATakingStudentUserID int not null,                  # references to student id

   AEvaluationID int not null,                              # references to eval id

   DueDate date,

   NumberOfAttempts int not null,

   CONSTRAINT pk_ASSIGNMENT primary key (ATakingStudentUserID, AEvaluationID),

   CONSTRAINT fk_ASSIGNMENT_STUDENT foreign key (ATakingStudentUserID) references
STUDENT(StudentUserID),

   CONSTRAINT fk_ASSIGNMENT_EVALUATION foreign key (AEvaluationID) references
EVALUATION(EvaluationID)

);


INSERT INTO ASSIGNMENT VALUES ("5", "2", "2022-01-31", "1");


DROP TABLE IF EXISTS QUIZ;

CREATE TABLE QUIZ(

        QTakingStudentUserID int not null,

   QEvaluationID int not null,

   CONSTRAINT pk_QUIZ primary key (QTakingStudentUserID, QEvaluationID),

   CONSTRAINT fk_QUIZ_STUDENT foreign key (QTakingStudentUserID) references STUDENT(StudentUserID),

   CONSTRAINT fk_QUIZ_EVALUATION foreign key (QEvaluationID) references EVALUATION(EvaluationID)

);


INSERT INTO QUIZ VALUES ("2", "1");

INSERT INTO QUIZ VALUES ("5", "2");


DROP TABLE IF EXISTS ENROLLS;

CREATE TABLE ENROLLS(

        EnrolledCourseCode int not null,
   EnrollingStudentUserID int not null,

   CONSTRAINT pk_ENROLLS primary key (EnrolledCourseCode, EnrollingStudentUserID),

   CONSTRAINT fk_ENROLLS_COURSE foreign key (EnrolledCourseCode) references COURSE(CourseCode),

   CONSTRAINT fk_ENROLLS_STUDENT foreign key (EnrollingStudentUserID) references
STUDENT(StudentUserID)

);


CREATE TRIGGER after_student_enrolls

AFTER INSERT ON ENROLLS

FOR EACH ROW

            UPDATE COURSE,ENROLLS SET StudentAmount = StudentAmount + 1 WHERE StudentAmount =
COURSE.StudentAmount


                                                AND ENROLLS.EnrolledCourseCode = COURSE.CourseCode;


INSERT INTO ENROLLS VALUES ("420", "5");

INSERT INTO ENROLLS VALUES ("420", "2");

INSERT INTO ENROLLS VALUES ("420", "0");

INSERT INTO ENROLLS VALUES ("538", "5");

INSERT INTO ENROLLS VALUES ("538", "2");


DROP TABLE IF EXISTS TAKE;

CREATE TABLE TAKE(

            TStudentUserID int not null,

   TEvaluationID int not null,

   EMakerID int not null,                                              # references teachingStaffID that makes
this evaluation

   EGraderID int not null,

   EvalTakenCourseID int not null,

   CONSTRAINT pk_TAKE primary key (TStudentUserID, TEvaluationID, EMakerID, EGraderID,
EvalTakenCourseID),

   CONSTRAINT fk_TAKE_STUDENT foreign key (TStudentUserID) references STUDENT(StudentUserID),

   CONSTRAINT fk_TAKE_EVALUATION foreign key (TEvaluationID) references EVALUATION(EvaluationID),
   CONSTRAINT fk_TAKE_TEACHING_STAFF1 foreign key (EMakerID) references
TEACHING_STAFF(TeachingUserID),

   CONSTRAINT fk_TAKE_TEACHING_STAFF2 foreign key (EGraderID) references
TEACHING_STAFF(TeachingUserID),

   CONSTRAINT fk_TAKE_COURSE foreign key (EvalTakenCourseID) references COURSE(CourseCode)

);


INSERT INTO TAKE VALUES ("2", "1", "9", "8", "420");

INSERT INTO TAKE VALUES ("2", "1", "8", "9", "538");

INSERT INTO TAKE VALUES ("5", "2", "6", "7", "538");


DROP TABLE IF EXISTS CERTIFICATE;

CREATE TABLE CERTIFICATE(

            CertificateID int not null,

   CertificateName varchar(25) not null,

   GivenDate date not null,

   CertificateCourseID int not null,                # references to courseCode

   CertificatedStudentUserID int not null,          # references to person that achieves certificate

   CertificatingEvaluationID int not null,          # references EvaluationID

   EMakerID int not null,                                          # references teachingStaffID that makes
this evaluation

   EGraderID int not null,                                         # references teachingStaffID that grades
this evaluation

            CONSTRAINT pk_CERTIFICATE primary key (CertificateID),

   CONSTRAINT fk_CERTIFICATE_COURSE foreign key (CertificateCourseID) references COURSE(CourseCode),

   CONSTRAINT fk_CERTIFICATE_STUDENT foreign key (CertificatedStudentUserID) references
STUDENT(StudentUserID),

            CONSTRAINT fk_CERTIFICATE_EVALUATION foreign key (CertificatingEvaluationID) references
EVALUATION(EvaluationID),

   CONSTRAINT fk_CERTIFICATE_TEACHING_STAFF1 foreign key (EMakerID) references
TEACHING_STAFF(TeachingUserID),

            CONSTRAINT fk_CERTIFICATE_TEACHING_STAFF2 foreign key (EGraderID) references
TEACHING_STAFF(TeachingUserID)

);
```

```
INSERT INTO CERTIFICATE VALUES ("41233", "Programming Cer.", "2018-04-22", "420", "5", "2", "7","8");
INSERT INTO CERTIFICATE VALUES ("40222", "Cer. of Education", "2008-04-22", "538", "2", "2", "9","6");

DROP TABLE IF EXISTS EARNS;
CREATE TABLE EARNS(
	EarningPersonUserID int not null,
   EarnedCertificateID int not null,
   CONSTRAINT pk_EARNS primary key (EarningPersonUserID, EarnedCertificateID),
   CONSTRAINT fk_EARNS_PERSON foreign key (EarningPersonUserID) references PERSON(PersonUserID),
   CONSTRAINT fk_EARNS_CERTIFICATE foreign key (EarnedCertificateID) references CERTIFICATE(CertificateID)
);

INSERT INTO EARNS VALUES("5", "41233");
INSERT INTO EARNS VALUES("2", "40222");

DROP TABLE IF EXISTS QUESTIONS;
CREATE TABLE QUESTIONS(
	QuestionID int not null,
   ExamEvaluationID int not null,                        # references evaluation
	QuestionType char not null,                                  # C for Classic, T for Test
   Answer text,
   CONSTRAINT pk_QUESTIONS primary key (QuestionID, ExamEvaluationID),
   CONSTRAINT fk_QUESTIONS_EVALUATION foreign key (ExamEvaluationID) references
EVALUATION(EvaluationID)
);

INSERT INTO QUESTIONS VALUES("711", "1", "C", "True");
INSERT INTO QUESTIONS VALUES("712", "2", "T", "D");

UPDATE TEACHER SET Salary = "40000.00" WHERE TeacherUserID = "6";
UPDATE COURSE SET CourseTeacherUserID = "9" WHERE CourseCode = "111";
UPDATE COMPANY SET CompanyName = "Red Bull Racing" WHERE CompanyUserID = "4";
```

```sql
/*SELECT * FROM TEACHER;

SELECT * FROM COURSE;

SELECT * FROM COMPANY;*/


DELETE FROM FEEDBACK WHERE FeedbackID = "15";

DELETE FROM JOB_ALERT WHERE AlertID = "999";

DELETE FROM POST WHERE PostID = "11";


SELECT * FROM FEEDBACK;

SELECT * FROM JOB_ALERT;

SELECT * FROM POST;


/*SELECT * FROM USERS;

SELECT * FROM PERSON;

SELECT * FROM COMPANY;

SELECT * FROM JOB_ALERT;

SELECT * FROM FORUM;

SELECT * FROM POST;

SELECT * FROM COMMENTS;

SELECT * FROM SAVES;

SELECT * FROM LIKES;

SELECT * FROM SKILL;

SELECT * FROM HOBBY;

SELECT * FROM TEACHING_STAFF;

SELECT * FROM STUDENT;

SELECT * FROM TEACHER;

SELECT * FROM ASSISTANT;

SELECT * FROM GRUP;

SELECT * FROM COURSE;

SELECT * FROM EVALUATION;*/
```

SQL EXAMPLES:

```
# teaching staff olan people
select *
from person
where personType = "Teaching Staff"
```

```
# londradaki is ilanlari
select *
from job_alert
where JobLocation = "London";
```

```
#ogrencilerin ortalamalari ve ID'leri
SELECT StudentUserID, GPA
FROM STUDENT
WHERE STUDENT.GPA>="2.50";
```

```
# diplomasini Ege Universitesinden alan ogretim uyeleri
select FName, LName, DiplomaGivenDate
from PERSON as p, TEACHING_STAFF as ts
where CollegeObtained = "Ege Univ" and TeachingUserID = PersonUserID;
```

```
# "17" PostID'li postu beğenen kisiler
SELECT P.PersonUserID, P.Fname, P.LName
FROM LIKES AS L, PERSON AS P
WHERE L.LikedPostID = "17" AND  L.LikingUserID = P.PersonUserID;
```

```
# sertifikasi olan ogrencilerin tum bilgileri
select *
from CERTIFICATE as c, PERSON as p
where CertificatedStudentUserID = PersonUserID;
```

```
# maasi en yuksek 3 hoca
```

```
SELECT  t1.Salary, PERSON.FName, PERSON.LName

FROM TEACHER as t1, PERSON

WHERE 3 >= (SELECT count(distinct Salary)

        FROM TEACHER as t2

        WHERE (t1.Salary <= t2.Salary) order by t1.Salary desc) AND t1.TeacherUserID = PERSON.PersonUserID;
```

```
# 420 numarali dersten ogrencilerin aldigi notlar, yuksekten aza siralanmis

select FName, LName, StudentUserID, Grade, CourseID

from PERSON, STUDENT, EVALUATION

where TakingStudentUserID = StudentUserID and StudentUserID = PersonUserID and CourseID = 420

order by Grade desc;
```

```
#Kisilerden WorkExperience'ı olanların calıstığı sirketler, calısanların isimleri ve baslama tarihleri

SELECT C.CompanyName, P.FName, P.LName, E.WorkingPosition, E.StartDate

FROM EXPERIENCE AS E, PERSON AS P, COMPANY AS C

WHERE E.ExperienceUserID = P.PersonUserID AND E.Company = C.CompanyUserID;
```

```
# content yukleyen ogretmenlerin listesi ve contentleri

select FName, LName, TeachingStaffType, ContentNo, FileType, UploadDate, DownDate

from CONTENT, TEACHING_STAFF, PERSON

where UploaderUserID = TeachingUserID and TeachingUserID = PersonUserID and TeachingStaffType = "T";
```

```
# 420 nolu dersin ogretmeni ile ilgili bilgiler

select *

from course, teacher, teaching_staff, person, users

where CourseCode = 420 and courseTeacherUserID = 8 and courseTeacherUserID = TeacherUserID

and teacherUserID = TeachingUserID and TeachingUserID = PersonUserID and PersonUserID = UserID;
```

```
# kisinin sinavina girdigi derslerden aldigi notlar

select FName, Minit, LName, Grade, CourseName

from evaluation, person, users, course

where TakingStudentUserID = 5 and TakingStudentUserID = PersonUserID and PersonUserID = UserID and
CourseCode = CourseID;
```

```
# kisilerin yetenek, hobi, ve sertifikalarini gosterio

select PersonUserID, FName, LName, Sex, CV, HobbyName, SkillName, CertificateName

from PERSON

LEFT JOIN HOBBY

ON(HobbyPersonUserID = PersonUserID)

LEFT JOIN SKILL

ON(SkillPersonUserID = PersonUserID)

LEFT JOIN CERTIFICATE

ON(CertificatedStudentUserID = PersonUserID);
```

```
# 1 kodlu dersten dersi gecen ogrenciler

select CourseID, TakingStudentUserID, FName, LName, Grade

from EVALUATION, STUDENT, PERSON, USERS

where EvaluationID = "1" and TakingStudentUserID = StudentUserID and StudentUserID = PersonUserID and
PersonUserID = UserID

                        and Grade >= 45

order by TakingStudentUserID asc;
```

```
select SendingStudentUserID, FeedbackID, Content, RecievingTeacherUserID, Major, FName as
RecievingTeacherFirstName, LName as  RecievingTeacherLastName

from STUDENT, FEEDBACK, TEACHER, PERSON

where SendingStudentUserID = StudentUserID and RecievingTeacherUserID = TeacherUserID and
TeacherUserID = PersonUserID;
```