EGE UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING

DATABASE MANAGEMENT

PROJECT REPORT

MADE BY

Tuğcan Topaloğlu

Elifnaz Acun

Simge Merve Yaşbay

Özlem Çelebi

Contents

1. Analysis 2
1.1 Brief Explanation 2
2.1 Analysis Report for Each Web application 2
2.1.a What is the aim of each application? 2
2.1.b What are the main entities of them? 2
2.1.c What are the characteristics of each entity? 2
2.1.d What relationships exists among the entities? 3
2.1.e What are the constraints related to entities, their characteristics and the relationships ssssamong them? 4
2. Design - Conceptual Design 5
3. Design - Logical Model 6
3.1 Mapping 6
3.2 Relational Model 9
4. Implementation - Physical Model 10
4.1 Tables 10
4.2 Tuples 20
4.3 Triggers 21
4.4 Check Constraints 23
4.5 SQL Statements 26
4.5.a INSERT/DELETE/UPDATE Statements 26
4.5.b Select Statements 27
4.5.c Original Select Statements 31

1. Analysis

1.1 Brief Explanation

If we look at these two programs independently for the first time, LinkedIn is a business networking application that connects registered individuals, corporations, and schools all around the world. Job advertising, employment applications, job contacts, and talents, work experiences, interests, and other fields relevant to each user are all included in the application, as are profiles in the form of a CV.

Moodle, on the other hand, caters to a more specialized audience and allows teachers to require students to register for the course using the passwords they have created, and it incorporates quizzes, exams, homework, and other course-related activities.

2.1 Analysis Report for Each Web application

2.1.a What is the aim of each application?

LinkedIn's goal is to link professionals all over the world in order to create a more efficient and productive business network by allowing them to interact. Moodle's goal is to provide a unique educational environment for educators and students through its use and control, resulting in a more efficient teaching platform.

2.1.b What are the main entities of them?

Common main entities of both Db_User , User_Profile , College , Faculty , Department , Transcript.

In addition to these, the main entities of LinkedIn Company , Company_Profile , Collection , Job_Offer , Post , Achievement.

In addition to these, the main entities of Moodle Course , Grading_Req.

2.1.c What are the characteristics of each entity?

Db_User entity has unique user_id,username and password. The user can be of Student,Teacher and Worker. It has flag attributes that control each of them. If the user is a worker it has sector,career title and work time. If the user is a teacher it has teacher degree.

User_Profile entity has unique user_profile_id , full name , unique mail , address , unique phone , sex and birth date.

Company entity entity has unique company_id and mgr_id.

Company_Profile entity has unique company_profile_id,unique company name,company location, and unique company phone.

Transcript entity has unique transcript no , type , taken date and gpa.

College entity has unique college id , college name , location and phone.

Faculty entity has unique faculty id , name , location and phone.

Department entity has unique id,phone , name and location.

Course entity has unique id,name and description.

Grading_Req entity has unique id and type.One Grading_req can be only one of four types.These are Quiz,Project,Exam and Homework.Each of them has grade attribute.

Achievement entity has unique id,type and date. One Achievement can be only one of four types. These are Test Score , Willing Project , Language and Certificate. Each of them has name attribute and Test Score has score attribute.

Collection entity has unique id. Collection includes two type data.These are Job Offer and Post. Job Offer has unique id , title and location. Post has unique id.

2.1.d What relationships exists among the entities?

Each Db_User must has a User_Profile.Each User_Profile must belong a Db_User.

Each User_Profile can display more than one profile, and each User_Profile can be viewed by more than one User_Profile.Each User_Profile can connect with multiple User_Profile.

Each Db_User can save more than one Collection. Each Collection must necessarily be registered by a Db_User.

Each Db_User can apply to or view more than one Job_Offer. Each Job_Offer can be applied or viewed by more than one Db_User.

Each Db_User can share, like, comment on more than one Post. Each Post can be liked, shared, commented on by more than one Db_User.

Each Db_User can achieve more than one Achievement. An Achievement can belong to more than one Db_User.

Each Db_User can upload more than one Private_File. One Private_File must necessarily be uploaded by a Db_User.

Each Db_User can has more than one Transcript. One Transcript can only belong to one Db_User.

One Transcript can taken by only one College. Each College can take more than one Transcript.

Each Student can study only one College. Each College can has more than one Student.

Each Teacher can work only one College. Each College can has more than one Teacher.

Each College can have more than one Faculty. Each Faculty can belong to only one College.

Each Faculty can has more than one Department. Each Department must necessarily and only belong to one Faculty.

Each Department can give one or more Course. Each Course is provided by only one Department.

Each Student can enroll in more than one Course. Each Course can receive more than one Student registration.

Each Teacher can teach more than one Course. Each Course must be taught by one or more Teacher.

Each Course must has one or more Grading_Req. Each Grading_Req must belong to only one Course.

Every Worker can has a career in a Company. Each Company can has more than one Worker.

Each Company must has a Company_Profile. Each Company_Profile must belong to a Company.

Each Company can offers more than one Job_Offer. Each Job_Offer must be offered by only one Company.

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

Db_User's user_id attiribute must be initialized between 100000 and 199999.

Transcript's transcprit_no attribute must be initialized between 200000 and 299999.

College's college_id attribute must be initialized between 300000 and 399999.

Faculty's faculty_id attribute must be initialized between 400000 and 499999.

Department's department_id attribute must be initialized between 500000 and 599999.

Course's course_id attribute must be initialized between 600000 and 699999.

Grading_Req's grading_req_id attribute must be initialized between 700000 and 799999.

Company's company_id attribute must be initialized between 800000 and 899999.

User_Profile's user_profile_id attribute must be initialized between 900000 and 999999.

Company_Profile's company_profile_id attribute must be initialized between 700000 and 799999.

Achievement's achievement_id attribute must be initialized between 110000 and 119999.

Collection's cllection_id attribute must be initialized between 120000 and 129999.

Job_Offer's job_offer_id attribute must be initialized between 130000 and 139999.

Post's post_id attribute must be initialized between 140000 and 149999.

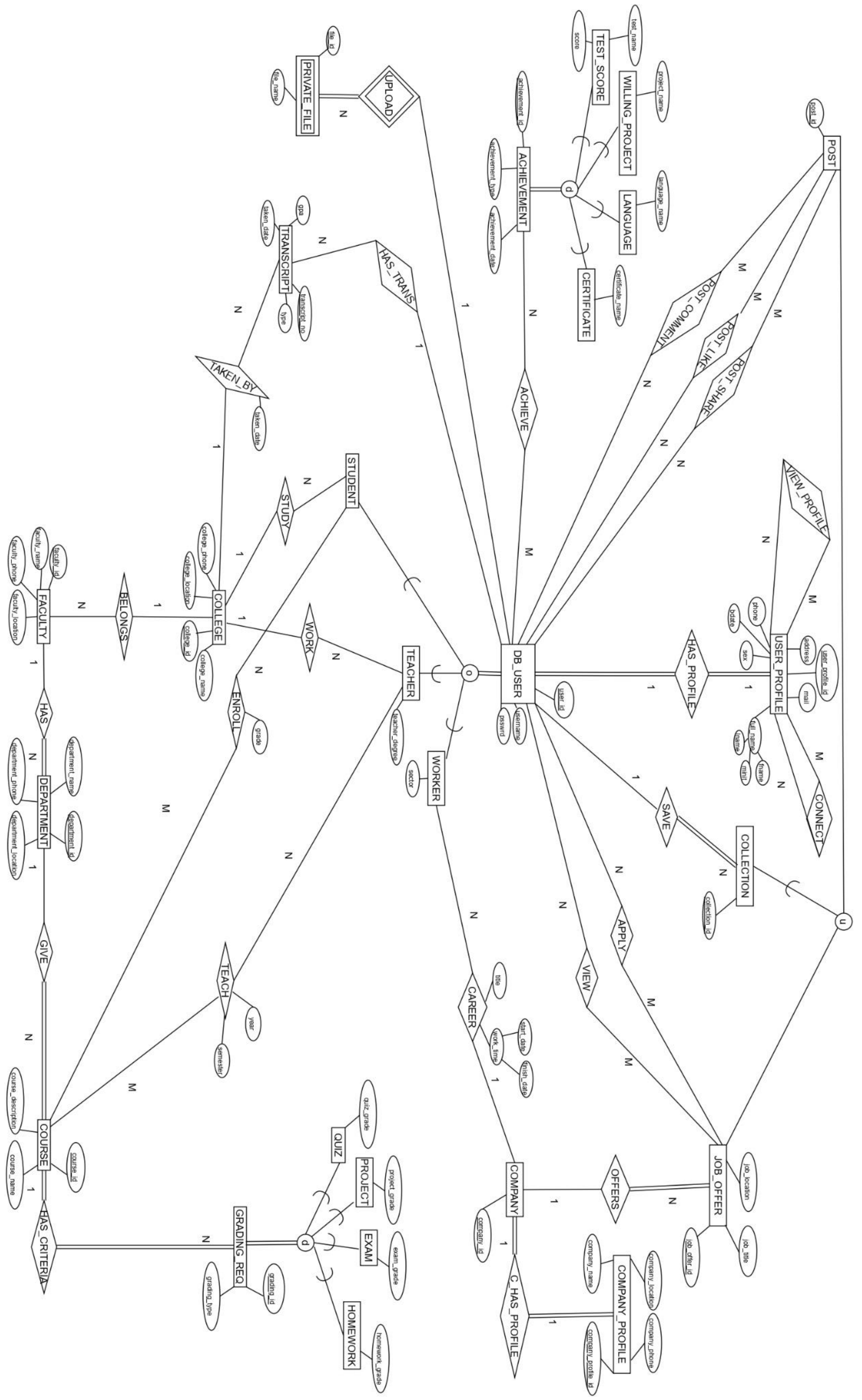
Private_File's file_id attribute must be initialized between 150000 and 159999.

Grading types can be Exa, Project, Homework or Quiz.

Achievements types can be Willing Project, Language, Test Score or Certificate.

Collection can be both job offer or post.

2. Design - Conceptual Design



3. Design - Logical Model

3.1 Mapping

1.ITERATION

1-)

TRANSCRIPT(transcript_no,type,taken_date,gpa)

COLLEGE(college_id,college_name,college_location,college_phone)

FACULTY(faculty_id,faculty_name,faculty_location,faculty_phone)

DEPARTMENT(department_id,department_phone,department_name,department_location)

COURSE(course_id,course_name,course_description)

COMPANY(company_id,mgr_id)

COMPANY_PROFILE(c_profile_id,company_name,comapny_loc,company_phone)

USER_PROFILE(user_profile_id,fname,minit,lname,mail,address,phone,sex,bdate)

2-)

3-) 1 to 1

COMPANY(company_id,mgr_id,c_profile_id) C_HAS_PROFILE

4-) 1 TO N

FACULTY(faculty_id,faculty_name,faculty_location,faculty_phone,faculty_college_id) BELONGS

DEPARTMENT(department_id,department_phone,department_name,department_location,department_faculty_id) HAS

COURSE(course_id,course_name,course_description,course_department_id) GIVE

TRANSCRIPT(transcript_no,type,taken_date,gpa,t_college_id) TAKEN_BY

5-) M to N

VIEW_PROFILE(user_profile_id,user_viewer_id)

CONNECT(user_profile_id,user_connect_id)

6-)

7-)

8-)

8.D.)

DB_USER(user_id,username,psswrd,student_flag,teacher_flag,worker_flag,teacher_degree,sector)

8.A.)

ACHIEVEMENT(achievemet_id,date,achievement_type)

TEST_SCORE(achievemet_id,score,test_name)

WILLING_PROJECT(achievemet_id,project_name)

LANGUAGE(achievement_id,language_name)

CERTIFICATE(achievement_id,certificate_name)

8.A.)

GRADING_REQ(grading_req_id,grading_type)

QUIZ(grading_req_id,quiz_grade)

PROJECT(grading_req_id,project_grade)

EXAM(grading_req_id,exam_grade)

HOMEWORK(grading_req_id,homework_grade)

9-)

COLLECTION(collection_id)

POST(post_id,post_collection_id)

JOB_OFFER(job_offer_id,job_collection_id,job_title,job_location)

2.ITERATION

1-)

2-)

PRIVATE_FILE(file_id,private_user_id,file_name) UPLOAD

3-) 1 to 1

DB_USER(user_id,username,psswrd,student_flag,teacher_flag,worker_flag,teacher_degree,sector,user_profile_id) HAS_PROFILE

4-) 1 to N

TRANSCRIPT(transcript_no,type,taken_date,gpa,t_college_id,t_user_id) HAS_TRANS

DB_USER(user_id,username,psswrd,student_flag,teacher_flag,worker_flag,teacher_degree,sector,user_profile_id,student_college_id) STUDY

DB_USER(user_id,username,psswrd,student_flag,teacher_flag,worker_flag,teacher_degree,sector,user_profile_id,student_college_id,teacher_college_id) WORK

DB_USER(user_id,username,psswrd,student_flag,teacher_flag,worker_flag,teacher_degree,sector,user_profile_id,student_college_id,teacher_college_id,worker_company_id,career_title,start_date,finish_date) CAREER

JOB_OFFER(job_offer_id,job_collection_id,job_title,job_location,job_offer_company_id) OFFERS

COLLECTION(collection_id,collection_user_id) SAVE

GRADING_REQ(grading_req_id,grading_type,grading_req_course_id) HAS_CRITERIA

5-) N TO M

LIKE(post_id,user_id)

COMMENT(post_id,user_id)

SHARE(post_id,user_id)

ENROLL(user_id,course_id,grade)

TEACH(user_id,course_id,year,semester)

ACHIEVE(achievement_id,user_id)

APPLY(job_offer_id,user_id)

VIEW(job_offer_id,user_id)

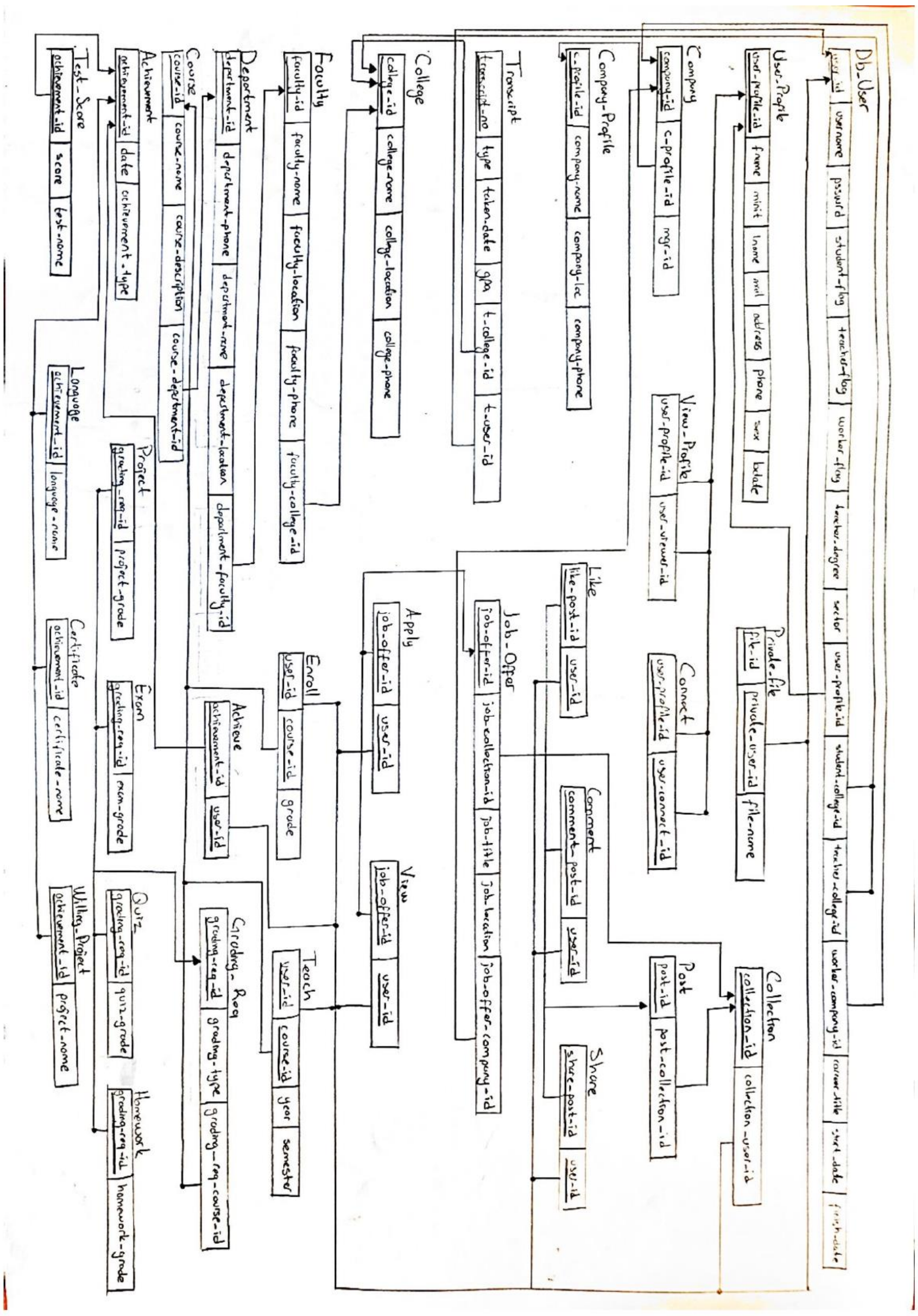
6-)

7-)

8-)

9-)

3.2 Relational Model


Db_User
user_id | username | pssword | student-flag | teacher-flag | worker-flag | teacher-degree | sector | user-profile-id | student-college-id | teacher-college-id | worker-company-id | career-title | start-date | finish-date
User-Profile
user-profile-id | fname | minit | lname | mail | address | phone | sex | bdate
Company
company-id | c-profile-id | mgr-id
Company-Profile
c-profile-id | company-name | company-loc | company-phone
Transcript
transcript-no | type | taken-date | gpa | t-college-id | t-user-id
College
college-id | college-name | college-location | college-phone
Faculty
faculty-id | faculty-name | faculty-location | faculty-phone | faculty-college-id
Department
department-id | department-phone | department-name | department-location | department-faculty-id
Course
course-id | course-name | course-description | course-department-id
Achievement
achievement-id | date | achievement-type
Test_Score
achievement_id | score | test-name
Language
achievement_id | language-name
Project
grading-req-id | project-grade
Certificate
achievement_id | certificate-name
Exam
grading-req-id | exam-grade
Achieve
achievement-id | user-id
Enroll
user-id | course-id | grade
Apply
job-offer-id | user-id
View
job-offer-id | user-id
Teach
user-id | course-id | year | semester
Grading_Req
grading-req-id | grading-type | grading-req-course-id
Quiz
grading-req-id | quiz-grade
Homework
grading-req-id | homework-grade
Willing_Project
achievement_id | project-name
View-Profile
user-profile-id | user-viewer-id
Like
like-post-id | user-id
Job-Offer
job-offer-id | job-collection-id | job-title | job-location | job-offer-company-id
Private-file
file-id | private-user-id | file-name
Connect
user-profile-id | user-connect_id
Comment
comment-post-id | user-id
Post
post-id | post-collection-id
Share
share-post-id | user-id
Collection
collection_id | collection-user-id

4. Implementation - Physical Model

Physical model implemented with POSTGRE SQL and pgAdmin 4.

All SQL codes can be found in Project folder. Tables can be found in “Tables.txt”. In order to maintain readibility all tuples are not included in tuples section and tuples can be found in “Tuples.txt”. Triggers can be found in “Triggers.txt”. Because of the disability of POSTGRE SQL instead of assertions we used check constraints and check constraints can be found in “CheckConstraints.txt”.Delete and Update insertions can be found in “DeleteUpdate.txt”. SQL statements can be found in “SelectStatements.txt”.

4.1 Tables

```
CREATE TYPE name AS(

Fname text,

Minit text,

Lname text

);


CREATE TYPE composite_time AS(

start_date Date,

finish_date Date

);


CREATE TABLE College(

college_id serial Primary Key NOT NULL,

college_name text unique NOT NULL,

college_location text,

college_phone integer unique

);


CREATE TABLE Company_Profile(

company_profile_id serial Primary Key NOT NULL,

company_name text UNIQUE NOT NULL,
company_location text,

company_phone integer UNIQUE

);

CREATE TABLE User_Profile(

user_profile_id serial Primary Key NOT NULL,

full_name name,

mail text unique NOT NULL,

address text,

phone integer unique,

sex character,

BDate Date

);

CREATE TABLE Company(

company_id serial Primary Key NOT NULL,

company_profile_id integer unique NOT NULL,

mgr_id integer unique NOT NULL,

Foreign Key(company_profile_id) references Company_Profile(company_profile_id) ON DELETE
CASCADE ON UPDATE CASCADE

);


CREATE TABLE Db_User(

user_id serial Primary Key NOT NULL,

username text unique NOT NULL,

psswrd integer NOT NULL,

student_flag boolean,

teacher_flag boolean,

worker_flag boolean,

sector text,
teacher_degree text,

student_college_id integer,

teacher_college_id integer ,

worker_company_id integer ,

user_profile_id integer unique NOT NULL,

career_title text,

work_time composite_time,

Foreign Key(student_college_id) references College (college_id) ON DELETE CASCADE ON UPDATE
CASCADE,

Foreign Key(teacher_college_id) references College (college_id) ON DELETE CASCADE ON UPDATE
CASCADE,

Foreign Key(worker_company_id) references Company (company_id) ON DELETE CASCADE ON
UPDATE CASCADE,

Foreign Key(user_profile_id) references User_Profile (user_profile_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Transcript(

transcript_no serial Primary Key NOT NULL,

Type text NOT NULL,

taken_date Date,

gpa numeric NOT NULL,

t_user_id integer NOT NULL,

t_college_id integer NOT NULL,

Foreign Key(t_user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(t_college_id) references College(college_id) ON DELETE CASCADE ON UPDATE CASCADE

);


CREATE TABLE Faculty (
faculty_id serial Primary Key NOT NULL,

faculty_name text NOT NULL,

faculty_location text,

faculty_phone integer unique,

faculty_college_id integer NOT NULL,

Foreign Key(faculty_college_id) references College(college_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Department (

department_id serial Primary Key NOT NULL,

department_faculty_id integer NOT NULL,

department_phone integer unique,

department_name text NOT NULL,

department_location text,

Foreign Key(department_faculty_id) references Faculty(faculty_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Course(

course_id serial Primary Key NOT NULL,

course_department_id integer NOT NULL,

course_name text NOT NULL,

course_description text,

Foreign Key(course_department_id) references Department(department_id) ON DELETE CASCADE
ON UPDATE CASCADE

);


CREATE TABLE Grading_Req (
);

grading_req_id serial Primary Key NOT NULL,

grading_req_course_id integer NOT NULL,

grading_type text NOT NULL,

Foreign Key(grading_req_course_id) references Course(course_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Quiz(

grading_req_id serial Primary Key NOT NULL,

quiz_grade integer NOT NULL,

Foreign Key(grading_req_id) references Grading_Req(grading_req_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Project(

grading_req_id serial Primary Key NOT NULL,

project_grade integer NOT NULL,

Foreign Key(grading_req_id) references Grading_Req(grading_req_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Exam(

grading_req_id serial Primary Key NOT NULL,

exam_grade integer NOT NULL,

Foreign Key(grading_req_id) references Grading_Req(grading_req_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Homework(
grading_req_id serial Primary Key NOT NULL,

homework_grade integer NOT NULL,

Foreign Key(grading_req_id) references Grading_Req(grading_req_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Private_File (

private_user_id integer NOT NULL,

file_id integer unique NOT NULL,

file_name text NOT NULL,

Primary Key(private_user_id, file_id),

Foreign Key(private_user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Achievement (

achievement_id serial Primary Key NOT NULL,

achievement_type text NOT NULL,

achievement_date Date

);


CREATE TABLE Willing_Project (

achievement_id serial Primary Key NOT NULL,

project_name text NOT NULL,

Foreign Key(achievement_id) references Achievement(achievement_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Language (
);

achievement_id serial Primary Key NOT NULL,

language_name text NOT NULL,

Foreign Key(achievement_id) references Achievement(achievement_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Certificate (

achievement_id serial Primary Key NOT NULL,

certificate_name text NOT NULL,

Foreign Key(achievement_id) references Achievement(achievement_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Test_Score (

achievement_id serial Primary Key NOT NULL,

test_name text NOT NULL,

score integer NOT NULL,

Foreign Key(achievement_id) references Achievement(achievement_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Collection (

collection_user_id integer NOT NULL,

collection_id  serial Primary Key NOT NULL,

Foreign Key(collection_user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Job_Offer (
job_offer_id serial Primary Key NOT NULL,

job_offer_company_id integer NOT NULL,

job_collection_id integer NOT NULL,

job_title text NOT NULL,

job_location text,

Foreign Key(job_collection_id) references Collection(collection_id) ON DELETE CASCADE ON UPDATE
CASCADE,

Foreign Key(job_offer_company_id) references Company(company_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Post (

post_id serial Primary Key NOT NULL,

post_collection_id integer NOT NULL,

Foreign Key(post_collection_id) references Collection(collection_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE View_Profile(

user_profile_id integer NOT NULL,

viewer_id integer NOT NULL,

Primary key (user_profile_id, viewer_id),

FOREIGN KEY(user_profile_id) REFERENCES User_Profile (user_profile_id) ON DELETE CASCADE ON
UPDATE CASCADE,

FOREIGN KEY(viewer_id) REFERENCES User_Profile (user_profile_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Connect(
);

user_profile_id integer NOT NULL,

user_connect_id integer NOT NULL,

Primary key (user_profile_id, user_connect_id),

FOREIGN KEY(user_profile_id) REFERENCES User_Profile (user_profile_id) ON DELETE CASCADE ON
UPDATE CASCADE,

FOREIGN KEY(user_connect_id) REFERENCES User_Profile (user_profile_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Enroll(

user_id integer NOT NULL,

course_id integer NOT NULL,

grade integer NOT NULL,

Primary Key(user_id, course_id),

FOREIGN KEY(user_id) REFERENCES Db_User (user_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(course_id) REFERENCES Course (course_id) ON DELETE CASCADE ON UPDATE CASCADE

);


CREATE TABLE Teach(

user_id integer NOT NULL,

course_id integer NOT NULL,

year integer,

semester text,

Primary Key(user_id, course_id),

FOREIGN KEY(user_id) REFERENCES Db_User (user_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(course_id) REFERENCES Course (course_id) ON DELETE CASCADE ON UPDATE CASCADE

);
```

```
CREATE TABLE Achieve(

user_id integer NOT NULL,

achievement_id integer NOT NULL,

Primary Key(user_id, achievement_id),

FOREIGN KEY(user_id) REFERENCES Db_User (user_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(achievement_id) REFERENCES Achievement (achievement_id) ON DELETE CASCADE ON
UPDATE CASCADE

);


CREATE TABLE Apply(

user_id integer NOT NULL,

job_offer_id integer NOT NULL,

Primary Key (user_id, job_offer_id),

Foreign Key(user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(job_offer_id) references Job_Offer(job_offer_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE View(

user_id integer NOT NULL,

job_offer_id integer NOT NULL,

Primary Key(user_id, job_offer_id),

Foreign Key(user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(job_offer_id) references Job_Offer(job_offer_id) ON DELETE CASCADE ON UPDATE
CASCADE

);


CREATE TABLE Post_Share(

post_id integer NOT NULL,
user_id integer NOT NULL,

Primary Key(post_id, user_id),

Foreign Key(post_id) references Post(post_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE

);


CREATE TABLE Post_Like(

post_id integer NOT NULL,

user_id integer NOT NULL,

Primary Key (post_id, user_id),

Foreign Key(post_id) references Post(post_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE

);


CREATE TABLE Post_Comment(

post_id integer NOT NULL,

user_id integer NOT NULL,

Primary Key (post_id, user_id),

Foreign Key(post_id) references Post(post_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user_id) references Db_User(user_id) ON DELETE CASCADE ON UPDATE CASCADE

);
```

4.2 Tuples

```
INSERT INTO Db_User VALUES
(100000,'worthbee',2154523,True,False,False,null,null,300000,null,null,900000,null,null);

INSERT INTO Db_User VALUES
(100001,'decayvowel',52312,True,True,False,null,'Undergraduate',300000,300001,null,900001,null,n
ull);

INSERT INTO Db_User VALUES
(100002,'pradawreck',54413312,False,False,True,'Advertisement',null,null,null,800000,900002,'Com
puter Engineer',('01-06-2019',null));
```

```
INSERT INTO Db_User VALUES
(100003,'cartierlinear',784515,True,True,True,'Banking','PHD',300000,300002,800001,900003,'Accou
nter',('16-09-2020','16-11-2020'));

INSERT INTO Db_User VALUES
(100004,'surroundgrind',214566,True,False,True,'Software',300001,null,300002,null,900004,'Human
Resources',('29-07-2021',null));

INSERT INTO Db_User VALUES
(100005,'jestawful',88962,True,False,False,null,null,300003,null,null,900005,null,null);

INSERT INTO Db_User VALUES
(100006,'frankfurtercovey',09066,False,True,False,null,'Graduate',null,300001,null,900006,null,null);

INSERT INTO Db_User VALUES
(100007,'spindlereverse',514812,False,False,True,'Hardware',null,null,null,800003,900007,'Software
Engineer',('03-01-2021','19-12-2021'));

INSERT INTO Db_User VALUES
(100008,'increasedrash',798900,False,True,True,'Banking','Proffesor',null,300001,800001,900008,'Se
curity',('24-04-2018',null));

INSERT INTO Db_User VALUES
(100009,'sowseforgetful',889521,True,True,True,'Marketing','PHD',300004,300001,800000,900009,'
Manager',('19-11-2017','02-09-2020'));

INSERT INTO Db_User VALUES
(100010,'sweepevery',2832521,False,False,True,'Hardware',null,null,null,800001,900010,'Manager',('
19-01-2018',null));

INSERT INTO Db_User VALUES
(100011,'founderwonder',5687521,False,False,True,'Banking',null,null,null,800002,900011,'Manager'
,('12-05-2016',null));

INSERT INTO Db_User VALUES
(100012,'phalangesexcited',5671521,False,False,True,'Marketing',null,null,null,800003,900012,'Mana
ger',('16-06-2015',null));
```

4.3 Triggers

```
CREATE FUNCTION gpa_controller()

RETURNS TRIGGER AS $gpa_controller$

   BEGIN

      --check that new gpa is  above 1,80

       IF NEW.gpa < 1.80 THEN

         RAISE EXCEPTION 'Transcript GPA can not be below 1,80.';
```

```
        END IF;

    END;

$gpa_controller$ LANGUAGE plpgsql;


CREATE TRIGGER trigger_gpa

BEFORE INSERT OR UPDATE OF gpa ON TRANSCRIPT

FOR EACH ROW

EXECUTE FUNCTION gpa_controller();


CREATE FUNCTION college_name()

RETURNS TRIGGER AS $college_name$

    BEGIN

        --check that college_name is given

        IF NEW.college_name IS NULL THEN

            RAISE EXCEPTION 'College Name cannot be null';

        END IF;

    END;

$college_name$ LANGUAGE plpgsql;


CREATE TRIGGER trigger_college_name

BEFORE INSERT OR UPDATE OF college_name ON COLLEGE

FOR EACH ROW

EXECUTE FUNCTION college_name();


CREATE FUNCTION psswrd_change()

RETURNS TRIGGER AS $psswrd_change$

    BEGIN
```

```
        --check that password has changed

        IF NEW.psswrd <> OLD.psswrd THEN

            RAISE EXCEPTION 'Password changed.';

        END IF;

        --check that new password is the same as the old one

        IF NEW.psswrd = OLD.psswrd THEN

            RAISE EXCEPTION 'New password can not be the same as the old one.';

        END IF;


    END;

$psswrd_change$ LANGUAGE plpgsql;


CREATE TRIGGER trigger_psswrd

BEFORE UPDATE OF psswrd ON DB_USER

FOR EACH ROW

EXECUTE FUNCTION psswrd_change();
```

4.4 Check Constraints

```
ALTER TABLE Db_User

ADD CONSTRAINT Db_User_CheckFinishDateGreaterThanStartDate

CHECK ((work_time).finish_date >= (work_time).start_date) ;


ALTER TABLE Quiz

ADD CONSTRAINT GRADE_CHECK

CHECK (quiz_grade>=40);


ALTER TABLE Project

ADD CONSTRAINT GRADE_CHECK

CHECK (project_grade>=40);
```

```
ALTER TABLE Exam

ADD CONSTRAINT GRADE_CHECK

CHECK (exam_grade>=40);


ALTER TABLE Homework

ADD CONSTRAINT GRADE_CHECK

CHECK (homework_grade>=40);


ALTER TABLE Achievement

ADD CONSTRAINT TYPE_CHECK_ACHIEVEMENT

CHECK (achievement_type IN ('Willing Project','Language','Test_Score','Certificate'));


ALTER TABLE Grading_Req

ADD CONSTRAINT TYPE_CHECK_REQ

CHECK (grading_type IN ('Quiz','Project','Exam','Homework'));


ALTER TABLE User_Profile

ADD CONSTRAINT CHECK_SEX

CHECK (Sex IN ('M','F'));


ALTER TABLE Db_User

ADD CONSTRAINT CHECK_LENGTH

CHECK (user_id Between 100000 and 199999);


ALTER TABLE Transcript

ADD CONSTRAINT CHECK_LENGTH
```

```
CHECK (transcript_no Between 200000 and 299999);

ALTER TABLE College
ADD CONSTRAINT CHECK_LENGTH
CHECK (college_id Between 300000 and 399999);

ALTER TABLE Faculty
ADD CONSTRAINT CHECK_LENGTH
CHECK (faculty_id Between 400000 and 499999);

ALTER TABLE Department
ADD CONSTRAINT CHECK_LENGTH
CHECK (department_id Between 500000 and 599999);

ALTER TABLE Course
ADD CONSTRAINT CHECK_LENGTH
CHECK (course_id Between 600000 and 699999);

ALTER TABLE Grading_Req
ADD CONSTRAINT CHECK_LENGTH
CHECK (grading_req_id Between 700000 and 799999);

ALTER TABLE Company
ADD CONSTRAINT CHECK_LENGTH
CHECK (company_id Between 800000 and 899999);

ALTER TABLE User_Profile
```

```
ADD CONSTRAINT CHECK_LENGTH

CHECK (user_profile_id Between 900000 and 999999);
```

4.5 SQL Statements

4.5.a INSERT/DELETE/UPDATE Statements

```
INSERT INTO Company_Profile VALUES(700000,'ARÇELİK','İZMİR',123665);

INSERT INTO Company_Profile VALUES(700001,'KOÇ','ANKARA',124785);

INSERT INTO Company_Profile VALUES(700002,'BEKO','İSTANBUL',96855);

INSERT INTO Company_Profile VALUES(700003,'ACUN MEDYA','İSTANBUL',127765);


DELETE FROM Db_User

WHERE username = 'worthbee';

DELETE FROM Faculty

WHERE faculty_college_id=2

DELETE FROM Company_Profile

WHERE company_location ='İZMİR'


UPDATE Db_User

SET sector='Marketing'

WHERE user_id = 100003

UPDATE Company_Profile

SET company_location ='ANKARA'

WHERE company_location='İSTANBUL'

UPDATE Achievement

SET achievement_date ='03-06-2022'

WHERE achievement_id=110016
```

4.5.b Select Statements

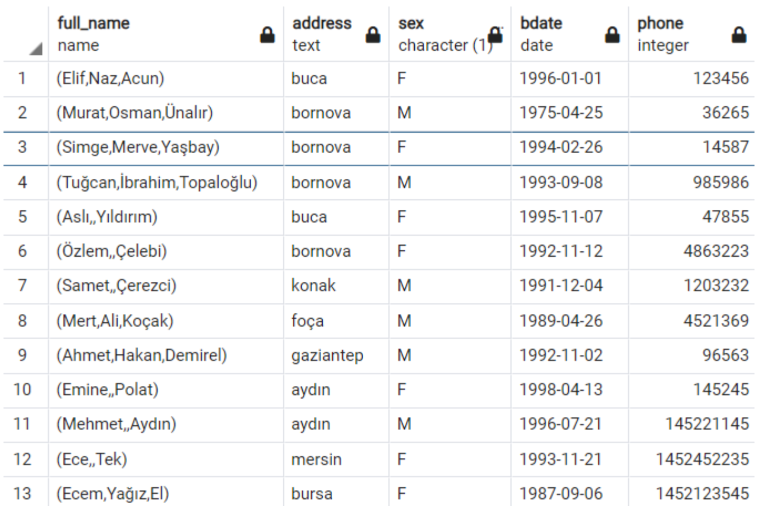
i) One Table

1)

Retrieve the full name, address, sex , birth date and phone of all users.

SELECT full_name,address,sex,BDate,phone

FROM User_Profile



	full_name name	address text	sex character (1)	bdate date	phone integer
1	(Elif,Naz,Acun)	buca	F	1996-01-01	123456
2	(Murat,Osman,Ünalır)	bornova	M	1975-04-25	36265
3	(Simge,Merve,Yaşbay)	bornova	F	1994-02-26	14587
4	(Tuğcan,İbrahim,Topaloğlu)	bornova	M	1993-09-08	985986
5	(Aslı,,Yıldırım)	buca	F	1995-11-07	47855
6	(Özlem,,Çelebi)	bornova	F	1992-11-12	4863223
7	(Samet,,Çerezci)	konak	M	1991-12-04	1203232
8	(Mert,Ali,Koçak)	foça	M	1989-04-26	4521369
9	(Ahmet,Hakan,Demirel)	gaziantep	M	1992-11-02	96563
10	(Emine,,Polat)	aydın	F	1998-04-13	145245
11	(Mehmet,,Aydın)	aydın	M	1996-07-21	145221145
12	(Ece,,Tek)	mersin	F	1993-11-21	1452452235
13	(Ecem,Yağız,El)	bursa	F	1987-09-06	1452123545

2)

Retrieve the user id and the user name of users who is either both student or teacher or both student or worker.

SELECT user_id,username

FROM Db_User

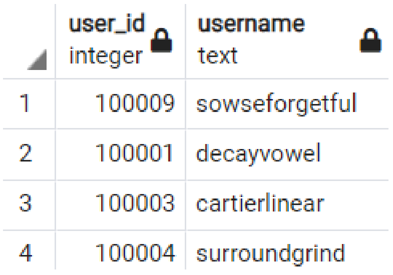
WHERE student_flag=true AND teacher_flag=true

UNION

SELECT user_id,username

FROM Db_User

WHERE student_flag=true AND worker_flag=true



	user_id integer	username text
1	100009	sowseforgetful
2	100001	decayvowel
3	100003	cartierlinear
4	100004	surroundgrind

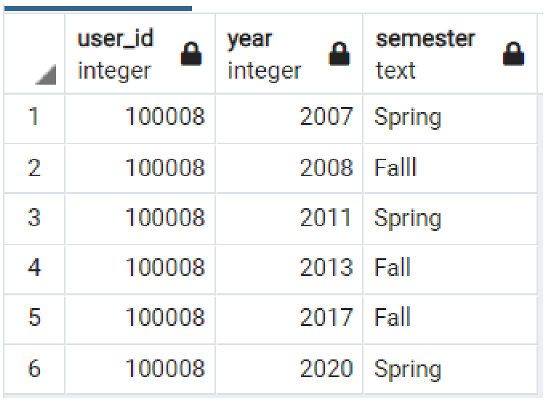
3)

Retrieve the year,semester of teacher with ID of 100008.

SELECT user_id,Teach.year,semester

FROM Teach

WHERE user_id=100008



	user_id integer	year integer	semester text
1	100008	2007	Spring
2	100008	2008	Falll
3	100008	2011	Spring
4	100008	2013	Fall
5	100008	2017	Fall
6	100008	2020	Spring

ii)Minimum Two Table

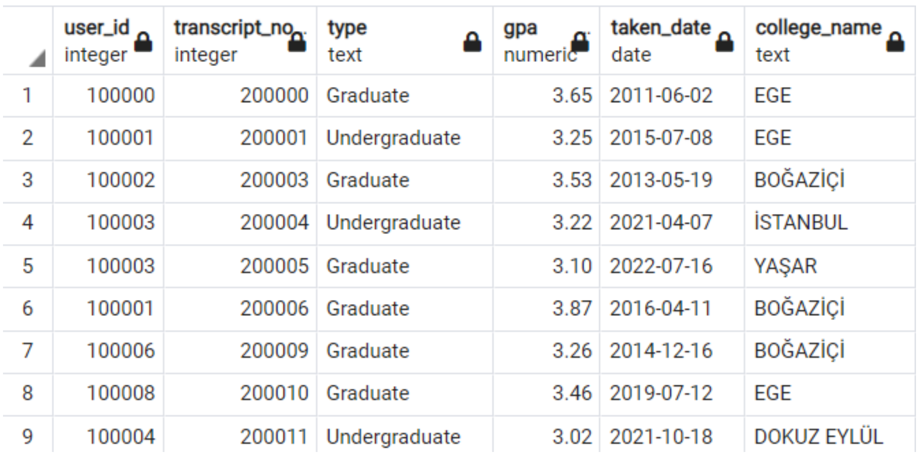
1)

Retrieve the users transcript information who has gpa more than 3.

SELECT user_id,transcript_no,type,gpa,taken_date,college_name

FROM Db_User,Transcript,College

WHERE gpa>3 AND t_user_id=user_id AND t_college_id=college_id



	user_id integer	transcript_no integer	type text	gpa numeric	taken_date date	college_name text
1	100000	200000	Graduate	3.65	2011-06-02	EGE
2	100001	200001	Undergraduate	3.25	2015-07-08	EGE
3	100002	200003	Graduate	3.53	2013-05-19	BOĞAZİÇİ
4	100003	200004	Undergraduate	3.22	2021-04-07	İSTANBUL
5	100003	200005	Graduate	3.10	2022-07-16	YAŞAR
6	100001	200006	Graduate	3.87	2016-04-11	BOĞAZİÇİ
7	100006	200009	Graduate	3.26	2014-12-16	BOĞAZİÇİ
8	100008	200010	Graduate	3.46	2019-07-12	EGE
9	100004	200011	Undergraduate	3.02	2021-10-18	DOKUZ EYLÜL

2)

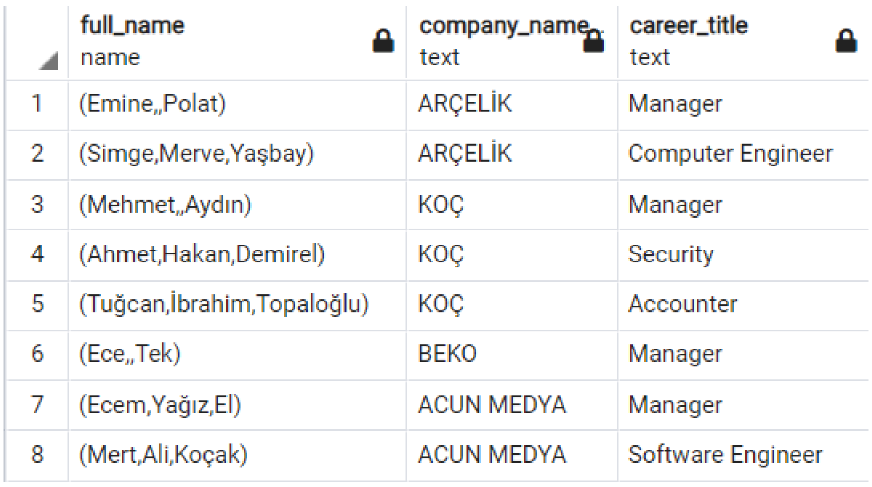
Retrieve the career title and company names of user who is worker .

```
SELECT full_name,company_name,career_title

FROM Db_User, Company, Company_Profile, User_Profile

WHERE worker_flag=true AND Db_User.user_profile_id=User_Profile.user_profile_id AND
worker_company_id=company_id

AND Company.company_profile_id=Company_Profile.company_profile_id
```



	full_name name	company_name text	career_title text
1	(Emine,,Polat)	ARÇELİK	Manager
2	(Simge,Merve,Yaşbay)	ARÇELİK	Computer Engineer
3	(Mehmet,,Aydın)	KOÇ	Manager
4	(Ahmet,Hakan,Demirel)	KOÇ	Security
5	(Tuğcan,İbrahim,Topaloğlu)	KOÇ	Accounter
6	(Ece,,Tek)	BEKO	Manager
7	(Ecem,Yağız,El)	ACUN MEDYA	Manager
8	(Mert,Ali,Koçak)	ACUN MEDYA	Software Engineer

3)

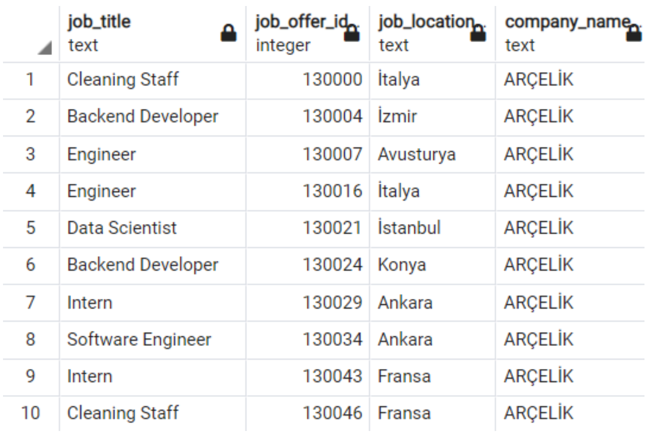
Retrieve the job offer information which offers by Arçelik.

```
SELECT job_title,job_offer_id,job_location,company_name

FROM Job_Offer, Company_Profile,Company

WHERE Company.company_id=800000 AND
Company.company_profile_id=Company_Profile.company_profile_id

AND Company.company_id=Job_Offer.job_offer_company_id
```



	job_title text	job_offer_id integer	job_location text	company_name text
1	Cleaning Staff	130000	İtalya	ARÇELİK
2	Backend Developer	130004	İzmir	ARÇELİK
3	Engineer	130007	Avusturya	ARÇELİK
4	Engineer	130016	İtalya	ARÇELİK
5	Data Scientist	130021	İstanbul	ARÇELİK
6	Backend Developer	130024	Konya	ARÇELİK
7	Intern	130029	Ankara	ARÇELİK
8	Software Engineer	130034	Ankara	ARÇELİK
9	Intern	130043	Fransa	ARÇELİK
10	Cleaning Staff	130046	Fransa	ARÇELİK

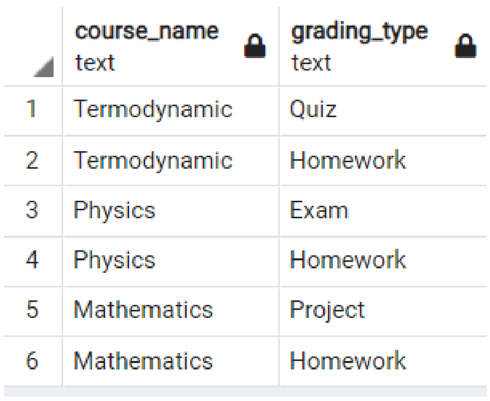
4)

Retrieve the grading requirements of the courses which in the department with ID of 500001

select course_name, grading_type

from Course, Grading_Req, Department

where department_id=500001 and course_department_id=department_id and course_id=grading_req_course_id



	course_name text	grading_type text
1	Termodynamic	Quiz
2	Termodynamic	Homework
3	Physics	Exam
4	Physics	Homework
5	Mathematics	Project
6	Mathematics	Homework

iii)Minimum Three Table

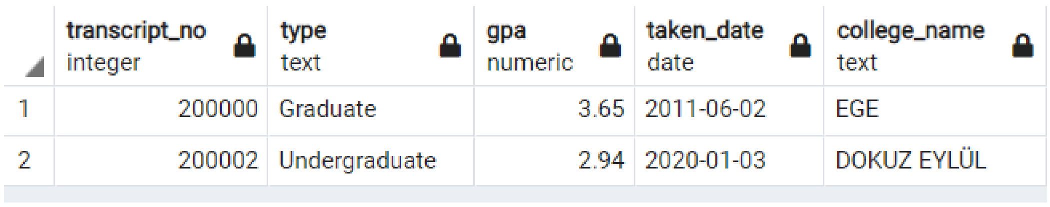
1)

Retrieve the transcript information of user with ID of 100000.

select transcript_no, Transcript.Type, gpa, taken_date, college_name

from Db_User, College, Transcript

where Db_user.user_id =100000 and t_user_id=user_id and t_college_id=college_id



	transcript_no integer	type text	gpa numeric	taken_date date	college_name text
1	200000	Graduate	3.65	2011-06-02	EGE
2	200002	Undergraduate	2.94	2020-01-03	DOKUZ EYLÜL

2)

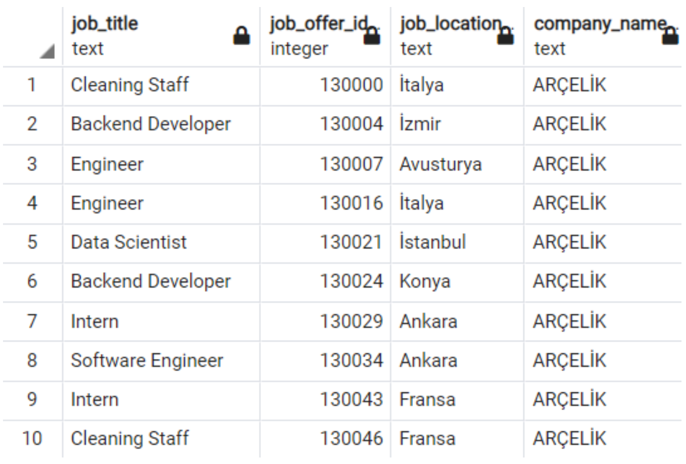
Retrieve the job offer information which offers by Arçelik.

SELECT job_title,job_offer_id,job_location,company_name

FROM Job_Offer, Company_Profile,Company

WHERE Company.company_id=800000 AND Company.company_profile_id=Company_Profile.company_profile_id

AND Company.company_id=Job_Offer.job_offer_company_id



	job_title text	job_offer_id integer	job_location text	company_name text
1	Cleaning Staff	130000	İtalya	ARÇELİK
2	Backend Developer	130004	İzmir	ARÇELİK
3	Engineer	130007	Avusturya	ARÇELİK
4	Engineer	130016	İtalya	ARÇELİK
5	Data Scientist	130021	İstanbul	ARÇELİK
6	Backend Developer	130024	Konya	ARÇELİK
7	Intern	130029	Ankara	ARÇELİK
8	Software Engineer	130034	Ankara	ARÇELİK
9	Intern	130043	Fransa	ARÇELİK
10	Cleaning Staff	130046	Fransa	ARÇELİK

3)

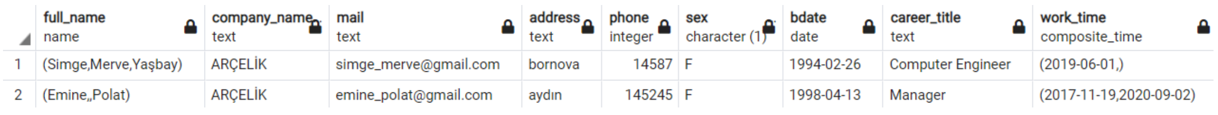
Retrieve the worker's informations who works at the company with id of 800000.

SELECT full_name,company_name,mail,address,phone,sex,BDate, career_title,work_time

FROM Db_User,User_Profile,Company,Company_Profile

WHERE company_id=800000 AND Db_User.worker_company_id=company_id AND Db_User.user_profile_id= User_Profile.user_profile_id

AND Company.company_profile_id=Company_Profile.company_profile_id



	full_name name	company_name text	mail text	address text	phone integer	sex character (1)	bdate date	career_title text	work_time composite_time
1	(Simge,Merve,Yaşbay)	ARÇELİK	simge_merve@gmail.com	bornova	14587	F	1994-02-26	Computer Engineer	(2019-06-01,)
2	(Emine,,Polat)	ARÇELİK	emine_polat@gmail.com	aydın	145245	F	1998-04-13	Manager	(2017-11-19,2020-09-02)

4.5.c Original Select Statements

1)

Retrieve the name of company which located in İzmir or offers a job in İzmir.

select distinct company_name

from Company_Profile

where company_name in (

select company_name

from Company_Profile

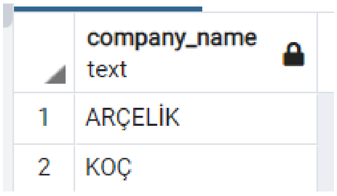
where company_location='İZMİR')

or

company_name in (select company_name

from Company_Profile, Job_Offer, Company

where job_location='İzmir' and job_offer_company_id=company_id and Company.company_profile_id=Company_Profile.company_profile_id)



	company_name text
1	ARÇELİK
2	KOÇ

2)

If the user having 900000 user ids has a connection in the company which has offer a job that the user applied for, it returns that contact's user id.

select user_id

from Db_User

where (worker_company_id) in (select company_id

from Apply, Db_User, Job_Offer, User_Profile, Company_Profile,Company

where User_Profile.user_profile_id=900000 and Db_User.user_id=Apply.user_id

and Apply.job_offer_id=Job_Offer.job_offer_id and Job_Offer.job_offer_company_id=Company.company_id)

intersect

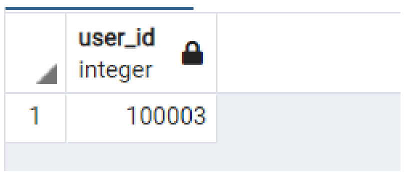
select user_id

from Db_User

where (user_profile_id) in (select user_connect_id

from Connect, Db_User, User_Profile

where User_Profile.user_profile_id=900000 and User_Profile.user_profile_id=Connect.user_profile_id)



	user_id integer
1	100003

3)

Returns the information of the teacher who teaches the courses in which the student with a user ID of 100000 is registered.

create view teacher(teacher_id, course_name, year, semester) as

(select distinct Teach.user_id, course_name, Teach.year, semester

from Enroll, Teach, Db_User, Course, User_Profile

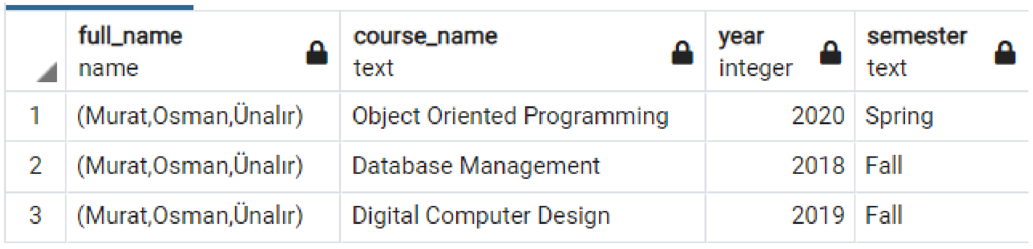
where Db_User.user_id=100000 and Db_User.user_id=Enroll.user_id and Enroll.course_id=Course.course_id

and Course.course_id=Teach.course_id)

select full_name, teacher.course_name, teacher.year, teacher.semester

from teacher, Db_User, User_Profile

where teacher.teacher_id=Db_User.user_id and Db_User.user_profile_id=User_Profile.user_profile_id



	full_name name	course_name text	year integer	semester text
1	(Murat,Osman,Ünalır)	Object Oriented Programming	2020	Spring
2	(Murat,Osman,Ünalır)	Database Management	2018	Fall
3	(Murat,Osman,Ünalır)	Digital Computer Design	2019	Fall

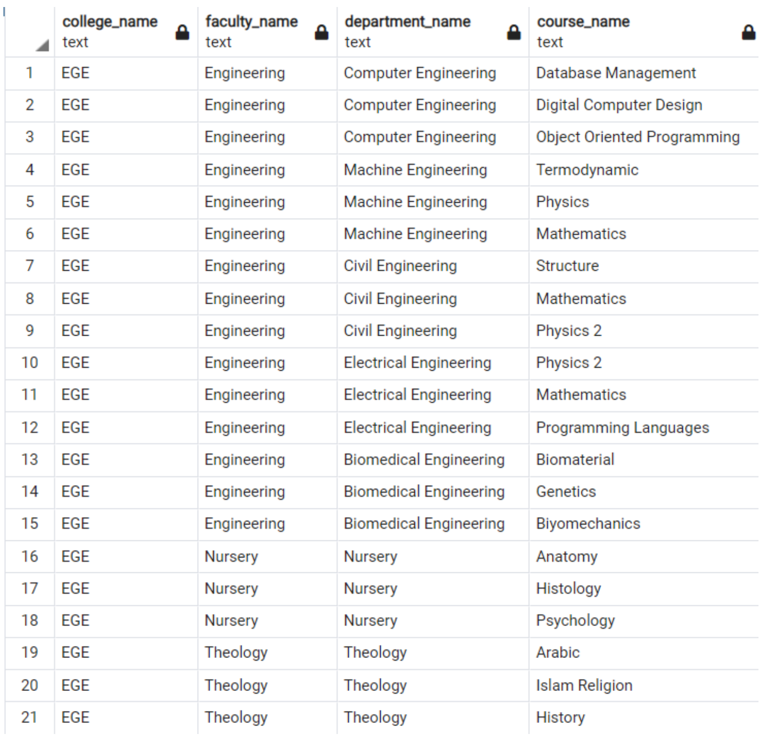
4)

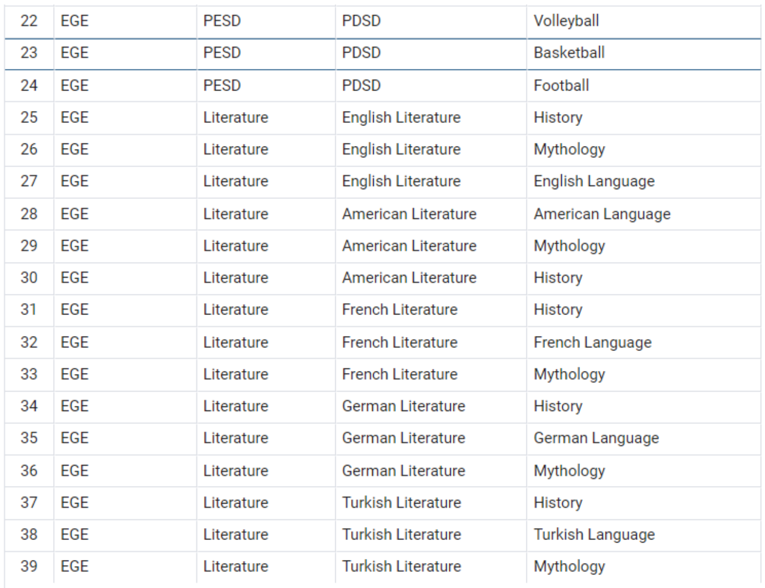
Retrieve the information of courses which has college_id of 300000

select college_name, faculty_name, department_name, course_name

from College, Faculty, Department, Course

where College.college_id=300000 AND College.college_id = Faculty.faculty_college_id AND Faculty.faculty_id=Department.department_faculty_id AND Department.department_id=Course.course_department_id





	college_name text	faculty_name text	department_name text	course_name text
1	EGE	Engineering	Computer Engineering	Database Management
2	EGE	Engineering	Computer Engineering	Digital Computer Design
3	EGE	Engineering	Computer Engineering	Object Oriented Programming
4	EGE	Engineering	Machine Engineering	Termodynamic
5	EGE	Engineering	Machine Engineering	Physics
6	EGE	Engineering	Machine Engineering	Mathematics
7	EGE	Engineering	Civil Engineering	Structure
8	EGE	Engineering	Civil Engineering	Mathematics
9	EGE	Engineering	Civil Engineering	Physics 2
10	EGE	Engineering	Electrical Engineering	Physics 2
11	EGE	Engineering	Electrical Engineering	Mathematics
12	EGE	Engineering	Electrical Engineering	Programming Languages
13	EGE	Engineering	Biomedical Engineering	Biomaterial
14	EGE	Engineering	Biomedical Engineering	Genetics
15	EGE	Engineering	Biomedical Engineering	Biyomechanics
16	EGE	Nursery	Nursery	Anatomy
17	EGE	Nursery	Nursery	Histology
18	EGE	Nursery	Nursery	Psychology
19	EGE	Theology	Theology	Arabic
20	EGE	Theology	Theology	Islam Religion
21	EGE	Theology	Theology	History
22	EGE	PESD	PDSD	Volleyball
23	EGE	PESD	PDSD	Basketball
24	EGE	PESD	PDSD	Football
25	EGE	Literature	English Literature	History
26	EGE	Literature	English Literature	Mythology
27	EGE	Literature	English Literature	English Language
28	EGE	Literature	American Literature	American Language
29	EGE	Literature	American Literature	Mythology
30	EGE	Literature	American Literature	History
31	EGE	Literature	French Literature	History
32	EGE	Literature	French Literature	French Language
33	EGE	Literature	French Literature	Mythology
34	EGE	Literature	German Literature	History
35	EGE	Literature	German Literature	German Language
36	EGE	Literature	German Literature	Mythology
37	EGE	Literature	Turkish Literature	History
38	EGE	Literature	Turkish Literature	Turkish Language
39	EGE	Literature	Turkish Literature	Mythology

5)

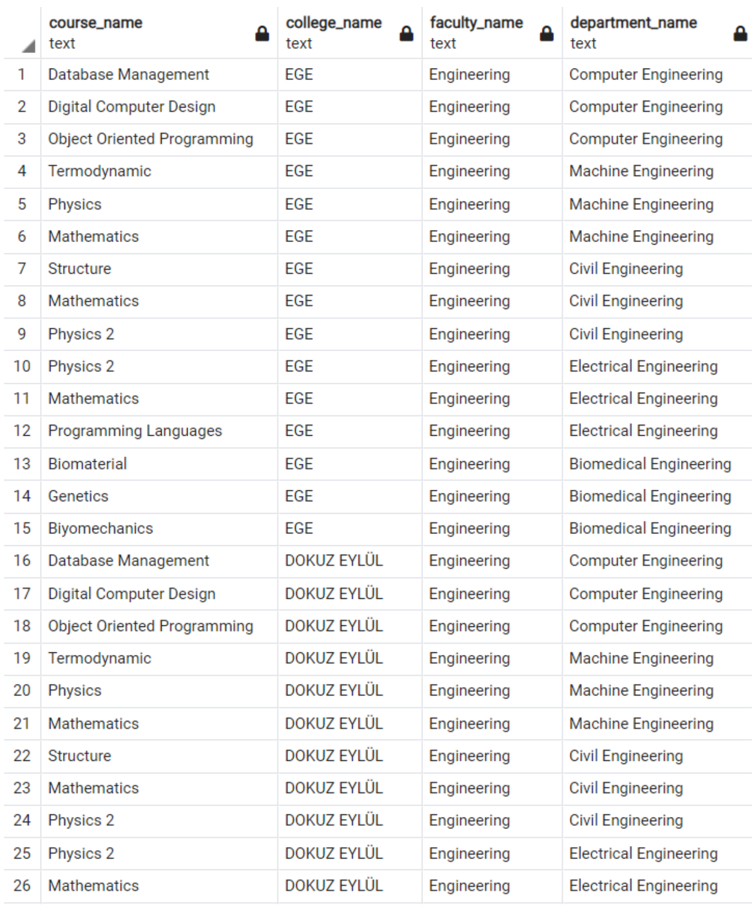
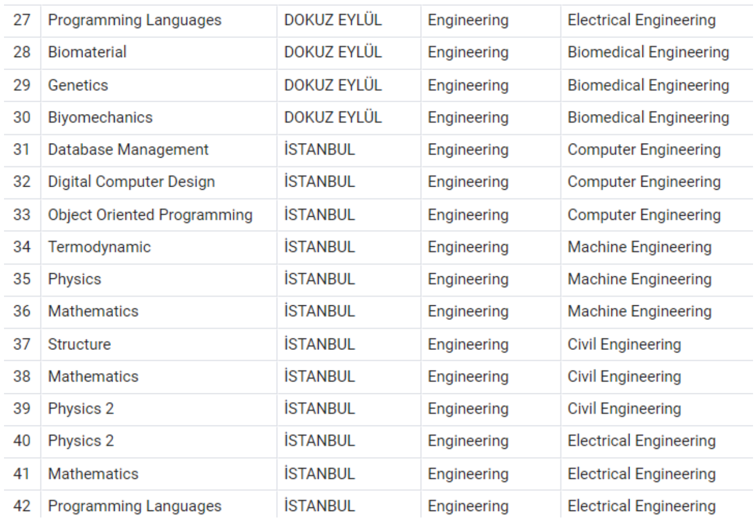
Retrive the course names and faculty names which contains "Engineering" in all colleges.

SELECT course_name,college_name,faculty_name,department_name

FROM College, Faculty, Department, Course

WHERE department_name LIKE '%Engineering' AND Department.department_faculty_id=Faculty.faculty_id

AND Faculty.faculty_college_id= College.college_id AND Department.department_id=Course.course_department_id

	course_name text	college_name text	faculty_name text	department_name text
1	Database Management	EGE	Engineering	Computer Engineering
2	Digital Computer Design	EGE	Engineering	Computer Engineering
3	Object Oriented Programming	EGE	Engineering	Computer Engineering
4	Termodynamic	EGE	Engineering	Machine Engineering
5	Physics	EGE	Engineering	Machine Engineering
6	Mathematics	EGE	Engineering	Machine Engineering
7	Structure	EGE	Engineering	Civil Engineering
8	Mathematics	EGE	Engineering	Civil Engineering
9	Physics 2	EGE	Engineering	Civil Engineering
10	Physics 2	EGE	Engineering	Electrical Engineering
11	Mathematics	EGE	Engineering	Electrical Engineering
12	Programming Languages	EGE	Engineering	Electrical Engineering
13	Biomaterial	EGE	Engineering	Biomedical Engineering
14	Genetics	EGE	Engineering	Biomedical Engineering
15	Biyomechanics	EGE	Engineering	Biomedical Engineering
16	Database Management	DOKUZ EYLÜL	Engineering	Computer Engineering
17	Digital Computer Design	DOKUZ EYLÜL	Engineering	Computer Engineering
18	Object Oriented Programming	DOKUZ EYLÜL	Engineering	Computer Engineering
19	Termodynamic	DOKUZ EYLÜL	Engineering	Machine Engineering
20	Physics	DOKUZ EYLÜL	Engineering	Machine Engineering
21	Mathematics	DOKUZ EYLÜL	Engineering	Machine Engineering
22	Structure	DOKUZ EYLÜL	Engineering	Civil Engineering
23	Mathematics	DOKUZ EYLÜL	Engineering	Civil Engineering
24	Physics 2	DOKUZ EYLÜL	Engineering	Civil Engineering
25	Physics 2	DOKUZ EYLÜL	Engineering	Electrical Engineering
26	Mathematics	DOKUZ EYLÜL	Engineering	Electrical Engineering
27	Programming Languages	DOKUZ EYLÜL	Engineering	Electrical Engineering
28	Biomaterial	DOKUZ EYLÜL	Engineering	Biomedical Engineering
29	Genetics	DOKUZ EYLÜL	Engineering	Biomedical Engineering
30	Biyomechanics	DOKUZ EYLÜL	Engineering	Biomedical Engineering
31	Database Management	İSTANBUL	Engineering	Computer Engineering
32	Digital Computer Design	İSTANBUL	Engineering	Computer Engineering
33	Object Oriented Programming	İSTANBUL	Engineering	Computer Engineering
34	Termodynamic	İSTANBUL	Engineering	Machine Engineering
35	Physics	İSTANBUL	Engineering	Machine Engineering
36	Mathematics	İSTANBUL	Engineering	Machine Engineering
37	Structure	İSTANBUL	Engineering	Civil Engineering
38	Mathematics	İSTANBUL	Engineering	Civil Engineering
39	Physics 2	İSTANBUL	Engineering	Civil Engineering
40	Physics 2	İSTANBUL	Engineering	Electrical Engineering
41	Mathematics	İSTANBUL	Engineering	Electrical Engineering
42	Programming Languages	İSTANBUL	Engineering	Electrical Engineering