**Database Management**

**2019-2020**

**Midterm Homework: LinkKariyerMood Database**

ANALYSIS

1. LinkedIn is a networking web site for businesses and people. Aim of this web site is to allow people and businesses create networks with each other. LinkedIn lets people to prepare a profile page where they can share their education and work experiences with other people. In LinkedIn people can follow businesses to be informed about changes in businesses or job offers. This website offers great convenience for community to share and know others’ experiences.

Kariyer.net is a web site for employees to search and apply for job offers and for companies to search and hire employees. Kariyer.net is apart from the LinkedIn is only created with the purpose of seeking a job. Kariyer.net lets employees to prepare CV which can be viewed by companies. Kariyer.net lacks the networking feature compared to Linkedn, but it makes it more centre on jobs and companies.

Moodle is a web site and a server for universities. Unlike the other web sites in this project, Moodle only focuses on education and supplying it. In Moodle universities can have seperate web pages for each of their departments. Each department’s page is composed of the course pages given in that department. In these course pages, teachers can share course materials or initialize homework submit areas where users can upload their homework files to the system. Moodle serves as a cloud storage so users of Moodle can upload and keep their files and access them anywhere.

1. A. LinkedIn aims connecting users in their each respectable line of work. Allowing them to build a network with people who are relevant to their career. It aims to be a professional networking site with social overtones.

Kariyer.net aims to be a bridge between many workplace who are in need of new employees and the people who can place this vacancies. It stores peoples Cvs and lets the business owner s see available candidates.

Moodle aims to be a learning platform which provides educators, administrators and learners with a secure, easy to access, integrated system to create learning environments.

B. Main entities of LinkedIn are; Member, Group, Company, Address,University.

Main entities of Kariyer.net are; Member, Skill, Address, Company, Job Offer, Office and University.

Main entities of Moodle are; Teacher, Student, Project, File, Faculty, Department, Course and University.

C. Each person has identifyng person number, mail address and non identifying, phone number, firstname, lastname and birthdate.

Each teacher has a branch of research field.

Each student has GPA indicating current point in university and grade implementing which and class the student is in.

Each member has a password to login to LinkKariyerMood.

Each skill has a identifying skill number and non identifying name, category.

Each address has an identifying address number and non identifying country, city, street information.

Each graduage level has an identifier number and non identifyng name.

Each department has an identifier number and non identifyng name.

Each faculty has an identifier number and non identifyng name.

Each course has identifyng course number and non identifying name, course code and credit.

Each project has identifyng project number and non identifyng title, description.

Each file has a name and privacy which indicates wheter file is public or private.

Each organization has identifyng organization number, mail address, name and non identifyng phone number.

Each university can have any number of faculties belonging to it.

Each group has identifyng group number and non identifyng name, description.

Each job offer has identifyng job offer number and non identifyng offer title, description.

Each office has identifyng office number and non identifyng name.

D. Each person must be any combination of teacher, student or member.

Each person can educate in any number of graduate levels.

Each person must live in exactly one address.

Each person can know any number of skills.

Each person can be referenced by any number of teachers.

Each teacher must work in a department.

Each teacher can message any number of students.

Each teacher can upload any number of files.

Each teacher can teach any number of courses.

Each student can upload any number of files.

Each student can work on any number of projects.

Each student can message any number of teachers.

Each student can enroll any number of courses.

Each member can recommend or be recommended by any number of members.

Each member can work for any number of offices.

Each member can follow any number of companies.

Each member can apply to any number of job offers.

Each member can create any number of groups.

Each member can join any number of groups.

Each member can connect with or be connected with any number of members.

Each skill can be known by any number of people.

Each address can be a residential addres of any number of people.

Each address can be a work address of any number of job offers.

Each address can be location of any number of offices.

Each graduate level can educate any number of people.

Each graduate level must be given by a department.

Each department can give any number of graduate levels.

Each department must belong to a faculty.

Each department must be located in a address.

Each department can give any number of courses.

Each faculty can have any number of departments.

Each faculty must belong to a university.

Each course can give any number of projects.

Each course must belong to a department.

Each course can be taught by any number of teachers.

Each course can be enrolled by any number of students.

Each project must be given by a course.

Each project can be worked on any number of students.

Each file can be upladed by a teacher or a student.

Each organization must be either one of the university or company.

Each university can have any number of faculties belonging to it.

Each group must be created by exactly one member.

Each group can have any number of members who joined it.

Each job offer must be given by an office.

Each job offer can be applied by any number of members.

Each job offer must have a work address.

Each company can have any number of offices.

Each company can be followed by any number of members.

Each office must be controlled by a company.

Each offica can give any number of job offers.

Each office must be located in an address.

Each office employs any number of members as employees.

E. For skill entity, skill’s category must be either one of Software, Finearts, Science or Sports

For person entity, person must be at least one or more of teacher, student and member.

For organization entity, type attribute must be one of CMP representing comapny or UNI representing university.

For office entity, the organization that office belongs to must be a company.

For faculty entity, the organization that faculty belongs to must be a university.

For student entity, the grade attribute must be one of 1, 2 ,3 or 4 representing which year the student is studying. For gpa attribute, gpa must be a float number in between 0 and 4.

For member entity, the password attribute must be equal or more than 8 digits for security purposes.

For file entity, the owner person of the file must belong to either student or teacher type.

For course entity, credit attribute must be greater than 0 and smaller than 9.

For any entites may have a start and end date attributes, start date must come before than end date.

DESIGN-LOGICAL MODEL

Iteration1

Step1

Skill(skill\_id, name, category)

Address(address\_id, country, city, street)

Group(group\_id, name, description)

Job\_offer(offer\_id, job\_title, description)

Office(office\_id, name)

File(file\_id, name, privacy)

Course(course\_id, name, code, credit)

Project(project\_id ,title, description)

Graduate\_level(grad\_id, name)

Department(dept\_id, name)

Faculty(faculty\_id, name)

Step2

-

Step3

-

Step4

Job\_offer(offer\_id, job\_title, description, address\_id, office\_id)

Project(project\_id ,title, description, course\_id)

File(file\_id, name, privacy, person\_id)

Course(course\_id, name, code, credit, dept\_id)

Project(project\_id ,title, description, course\_id)

Graduate\_level(grad\_id, name, dept\_id)

Department(dept\_id, name, grad\_id, address\_id)

Step 5

-

Step 6

-

Step 7

-

Step 8

Person(person\_id, fname, lname, phone, mail, bday)

Teacher(teacher\_id, branch)

Student(student\_id, gpa, grade)

Member(member\_id, password)

Organization(org\_id, name, phone, mail, type)

Step 9

-

Iteration 2

Step1

-

Step2

-

Step 3

-

Step 4

Person(person\_id, fname, lname, phone, mail, bday,type, address\_id)

Teacher(teacher\_id, branch, dept\_id)

Group(group\_id, name, description, member\_id)

Office(office\_id, name, address\_id, org\_id)

Faculty(faculty\_id, name, org\_id)

Step 5

Educates\_in(start\_date, end\_date, gpa, grad\_id, person\_id)

Messages(from\_person\_id, to\_person\_id, title, context,date)

References(teacher\_id, referenced\_person\_id, date, context)

Knows(person\_id, skill\_id)

Teaches(teacher\_id, course\_id, semester)

Enrolls(student\_id, course\_id, semester, grade)

Works\_on(student\_id, project\_id, grade)

Recommends(member\_id, recommended\_person\_id, date, context)

Follows(member\_id, org\_id)

Works\_for(member\_id, start\_date, end\_date, job\_title, office\_id)

Applies(member\_id, status, date, address\_id, offer\_id)

Joins(member\_id, group\_id)

Connects(member\_id, connected\_member\_id)

Step 6

-

Step7

-

Step8

-

Step9

-

IMPLEMENTATION-PHYSICAL MODEL

1. CREATION SCRIPT

CREATE TABLE tbl\_skill (

skill\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL UNIQUE,

category TEXT NOT NULL,

CHECK(category IN ('Software', 'FineArts', 'Science','Sports'))

);

Create TABLE tbl\_address(

address\_id SERIAL PRIMARY KEY,

country TEXT NOT NULL,

city TEXT NOT NULL,

street TEXT NOT NULL,

UNIQUE(country, city, street)

);

CREATE TABLE tbl\_organization(

org\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL UNIQUE,

phone VARCHAR(11) NOT NULL,

mail TEXT NOT NULL UNIQUE,

type VARCHAR(3) NOT NULL,

CHECK( type IN ('CMP','UNI'))

);

CREATE FUNCTION get\_type(data\_id INT)

RETURNS VARCHAR(3)

AS $$

BEGIN

RETURN (SELECT type FROM tbl\_organization WHERE org\_id = data\_id);

END; $$

LANGUAGE PLPGSQL;

CREATE FUNCTION assert\_is\_teacher(data\_id INT)

RETURNS VARCHAR(5)

AS $$

BEGIN

IF ( SELECT EXISTS (SELECT 1 FROM tbl\_teacher WHERE teacher\_id = data\_id)) THEN

RETURN 'True';

ELSE

RETURN 'False';

END IF;

END; $$

LANGUAGE PLPGSQL;

CREATE FUNCTION assert\_is\_student(data\_id INT)

RETURNS VARCHAR(5)

AS $$

BEGIN

IF ( SELECT EXISTS (SELECT 1 FROM tbl\_student WHERE student\_id = data\_id)) THEN

RETURN 'True';

ELSE

RETURN 'False';

END IF;

END; $$

LANGUAGE PLPGSQL;

CREATE FUNCTION assert\_is\_member(data\_id INT)

RETURNS VARCHAR(5)

AS $$

BEGIN

IF ( SELECT EXISTS (SELECT 1 FROM tbl\_member WHERE member\_id = data\_id)) THEN

RETURN 'True';

ELSE

RETURN 'False';

END IF;

END; $$

LANGUAGE PLPGSQL;

CREATE TABLE tbl\_office(

office\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

org\_id INTEGER NOT NULL REFERENCES tbl\_organization(org\_id) ON DELETE CASCADE,

address\_id INTEGER NOT NULL REFERENCES tbl\_address(address\_id) ON DELETE CASCADE,

CHECK( 'CMP' = get\_type(org\_id)),

UNIQUE(org\_id ,name, address\_id)

);

CREATE TABLE tbl\_job\_offer(

offer\_id SERIAL PRIMARY KEY,

job\_title TEXT NOT NULL,

description TEXT NOT NULL,

address\_id INTEGER NOT NULL REFERENCES tbl\_address(address\_id) ON DELETE CASCADE,

office\_id INT NOT NULL REFERENCES tbl\_office(office\_id) ON DELETE CASCADE,

UNIQUE (job\_title, address\_id, office\_id)

);

CREATE TABLE tbl\_faculty(

faculty\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

org\_id INTEGER NOT NULL REFERENCES tbl\_organization(org\_id) ON DELETE CASCADE,

CHECK( 'UNI' = get\_type(org\_id)),

UNIQUE (org\_id , name)

);

CREATE TABLE tbl\_department(

dept\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

faculty\_id INT NOT NULL REFERENCES tbl\_faculty(faculty\_id) ON DELETE CASCADE,

address\_id INTEGER NOT NULL REFERENCES tbl\_address(address\_id) ON DELETE CASCADE,

UNIQUE (dept\_id , name)

);

CREATE TABLE tbl\_graduate\_level(

grad\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

dept\_id INT NOT NULL REFERENCES tbl\_department(dept\_id) ON DELETE CASCADE,

UNIQUE (grad\_id , name)

);

CREATE TABLE tbl\_person(

person\_id SERIAL PRIMARY KEY,

fname TEXT NOT NULL,

lname TEXT NOT NULL,

phone VARCHAR(11),

mail TEXT NOT NULL UNIQUE,

bday DATE NOT NULL,

address\_id INTEGER NOT NULL REFERENCES tbl\_address(address\_id) ON DELETE CASCADE

);

CREATE TABLE tbl\_teacher(

teacher\_id INT NOT NULL UNIQUE REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

branch TEXT,

dept\_id INT NOT NULL REFERENCES tbl\_department(dept\_id) ON DELETE CASCADE

);

CREATE TABLE tbl\_student(

student\_id INT NOT NULL UNIQUE REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

gpa FLOAT,

grade varchar(1) NOT NULL,

CHECK( grade IN ('1', '2', '3', '4')),

CHECK( gpa = null OR ( 0<= gpa AND gpa <= 4.0) )

);

CREATE TABLE tbl\_member(

member\_id INT UNIQUE REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

password TEXT NOT NULL,

CHECK( char\_length(password)>=8 )

);

CREATE TABLE tbl\_file(

file\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

privacy BOOLEAN DEFAULT TRUE,

person\_id INTEGER NOT NULL REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

CHECK( 'True' = assert\_is\_student(person\_id) OR 'True' = assert\_is\_teacher(person\_id)),

UNIQUE(person\_id, name)

);

CREATE TABLE tbl\_course(

course\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

code TEXT NOT NULL,

credit INTEGER NOT NULL,

dept\_id INT NOT NULL REFERENCES tbl\_department(dept\_id) ON DELETE CASCADE,

CHECK(0<credit AND credit<9)

);

CREATE TABLE tbl\_project(

project\_id SERIAL PRIMARY KEY,

title TEXT NOT NULL,

description TEXT NOT NULL,

course\_id INTEGER NOT NULL REFERENCES tbl\_course(course\_id) ON DELETE CASCADE,

UNIQUE(course\_id , title)

);

CREATE TABLE tbl\_group(

group\_id SERIAL PRIMARY KEY,

name TEXT NOT NULL UNIQUE,

description TEXT NOT NULL,

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE

);

CREATE TABLE tbl\_educates\_in(

start\_date DATE NOT NULL,

end\_date DATE,

gpa FLOAT,

grad\_id INT NOT NULL REFERENCES tbl\_graduate\_level(grad\_id) ON DELETE CASCADE,

person\_id INT NOT NULL REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

CHECK(gpa = null OR ( 0<= gpa AND gpa <= 4.0)),

CHECK( start\_date < end\_date ),

UNIQUE(grad\_id , person\_id)

);

CREATE TABLE tbl\_references(

teacher\_id INTEGER NOT NULL REFERENCES tbl\_teacher(teacher\_id) ON DELETE CASCADE,

referenced\_person\_id INTEGER NOT NULL REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

date DATE NOT NULL,

context TEXT NOT NULL,

CHECK(teacher\_id != referenced\_person\_id),

UNIQUE(teacher\_id ,referenced\_person\_id, date)

);

CREATE TABLE tbl\_knows(

person\_id INT REFERENCES tbl\_person(person\_id) ON DELETE CASCADE,

skill\_id INT REFERENCES tbl\_skill(skill\_id) ON DELETE CASCADE,

UNIQUE(person\_id , skill\_id)

);

CREATE TABLE tbl\_teaches(

teacher\_id INTEGER NOT NULL REFERENCES tbl\_teacher(teacher\_id) ON DELETE CASCADE,

course\_id INTEGER NOT NULL REFERENCES tbl\_course(course\_id) ON DELETE CASCADE,

semester TEXT NOT NULL,

UNIQUE(teacher\_id, course\_id, semester)

);

CREATE TABLE tbl\_messages(

from\_person\_id INT NOT NULL REFERENCES tbl\_person(person\_id),

to\_person\_id INT NOT NULL REFERENCES tbl\_person(person\_id),

title TEXT,

context TEXT,

date DATE NOT NULL,

CHECK( ('True' = assert\_is\_student(to\_person\_id) OR 'True' = assert\_is\_teacher(from\_person\_id)) OR

( ('True' = assert\_is\_student(from\_person\_id) OR 'True' = assert\_is\_teacher(to\_person\_id))),

CHECK(from\_person\_id != tpo\_person\_id)

);

CREATE TABLE tbl\_works\_on(

student\_id INTEGER NOT NULL REFERENCES tbl\_student(student\_id) ON DELETE CASCADE,

project\_id INTEGER NOT NULL REFERENCES tbl\_project(project\_id) ON DELETE CASCADE,

grade INT,

semester TEXT NOT NULL,

UNIQUE(student\_id, project\_id, semester),

CHECK( grade = null OR ( 0<= grade AND grade<= 100))

);

CREATE TABLE tbl\_recommends(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

recommended\_member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

date DATE NOT NULL,

context TEXT NOT NULL,

CHECK(member\_id != recommended\_member\_id),

UNIQUE(member\_id, recommended\_member\_id, context )

);

CREATE TABLE tbl\_follows(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

org\_id INTEGER NOT NULL REFERENCES tbl\_organization(org\_id) ON DELETE CASCADE,

CHECK( 'CMP' = get\_type(org\_id)),

UNIQUE(member\_id , org\_id)

);

CREATE TABLE tbl\_enrolls(

student\_id INT NOT NULL REFERENCES tbl\_student(student\_id) ON DELETE CASCADE,

course\_id INT NOT NULL REFERENCES tbl\_course(course\_id) ON DELETE CASCADE,

semester TEXT NOT NULL,

grade FLOAT,

CHECK (grade = null OR ( 0<= grade AND grade <= 4.0)),

UNIQUE(student\_id, course\_id ,semester)

);

CREATE TABLE tbl\_works\_for(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

start\_date DATE NOT NULL,

end\_date DATE,

job\_title TEXT NOT NULL,

office\_id INT NOT NULL REFERENCES tbl\_office(office\_id) ON DELETE CASCADE,

CHECK( start\_date < end\_date )

);

CREATE TABLE tbl\_applies(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

date DATE NOT NULL,

status TEXT DEFAULT 'NOTEXAMINED',

offer\_id INT NOT NULL REFERENCES tbl\_job\_offer(offer\_id) ON DELETE CASCADE,

CHECK(status IN ('NOTEXAMINED', 'REJECTED', 'ACCEPTED')),

UNIQUE(offer\_id , member\_id)

);

CREATE TABLE tbl\_joins(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

group\_id INTEGER NOT NULL REFERENCES tbl\_group(group\_id) ON DELETE CASCADE,

UNIQUE(member\_id, group\_id)

);

CREATE TABLE tbl\_connects(

member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

connected\_member\_id INTEGER NOT NULL REFERENCES tbl\_member(member\_id) ON DELETE CASCADE,

CHECK(member\_id != connected\_member\_id),

UNIQUE(member\_id,connected\_member\_id )

);

CREATE OR REPLACE PROCEDURE add\_person\_student(

firstname text,

lastname text,

phone varchar(11),

email text,

bday DATE,

address\_id INT,

gpa FLOAT,

grade varchar(1))

LANGUAGE 'plpgsql'

AS $$

BEGIN

WITH new\_student as (

INSERT INTO public.tbl\_person(fname, lname, phone, mail, bday, address\_id)

VALUES ( firstname, lastname, phone, email, bday, address\_id) returning person\_id )

INSERT INTO tbl\_student(student\_id, gpa, grade) SELECT person\_id,gpa, grade FROM new\_student ;

COMMIT;

END;

$$;

CREATE OR REPLACE PROCEDURE add\_person\_member(

firstname text,

lastname text,

phone varchar(11),

email text,

bday DATE,

address\_id INT,

password TEXT)

LANGUAGE 'plpgsql'

AS $$

BEGIN

WITH new\_member as (

INSERT INTO public.tbl\_person(fname, lname, phone, mail, bday, address\_id)

VALUES ( firstname, lastname, phone, email, bday, address\_id) returning person\_id )

INSERT INTO tbl\_member(member\_id, password) SELECT person\_id ,password FROM new\_member;

COMMIT;

END;

$$;

CREATE OR REPLACE PROCEDURE add\_person\_teacher(

firstname text,

lastname text,

phone varchar(11),

email text,

bday DATE,

address\_id INT,

branch TEXT,

dept\_id)

LANGUAGE 'plpgsql'

AS $$

BEGIN

WITH new\_teacher as (

INSERT INTO public.tbl\_person(fname, lname, phone, mail, bday, address\_id)

VALUES ( firstname, lastname, phone, email, bday, address\_id) returning person\_id )

INSERT INTO tbl\_teacher(teacher\_id, branch, dept\_id) SELECT person\_id , branch , dept\_id FROM new\_teacher ;

COMMIT;

END;

$$;

1. INSERTION SCRIPT

INSERT INTO public.tbl\_skill(

name, category)

VALUES ( '3D art skills', 'FineArts'),

( 'Quantum Phsyics knowledge', 'Science'),

( 'Python knowledge', 'Software');

INSERT INTO public.tbl\_address(

country, city, street)

VALUES ( 'Turkey', 'Istanbul', 'Cumhuriyet Street'),

( 'Turkey', 'Bursa', 'Zafer Street'),

( 'Turkey', 'Izmir', 'Sencer Street'),

( 'Turkey', 'Izmir', 'Ataturk Street');

INSERT INTO public.tbl\_organization(

name, phone, mail, type)

VALUES ( 'Ege Üniversitesi', 2323881032, 'webadmin@ege.edu.tr', 'UNI'),

('9 Eylül Üniversitesi', 2324121212, 'egitim@deu.edu.tr', 'UNI'),

('Ege Profil', 2323989898, 'info@egeprofil.com.tr', 'CMP'),

('İzmir Yazilim', 2323453190, 'info@izmiryazilim.com.tr', 'CMP');

INSERT INTO public.tbl\_office(

name, org\_id, address\_id)

VALUES ('IT', 3,2),

('Frontend', 4,1),

('R&D', 3,2);

INSERT INTO public.tbl\_job\_offer(

job\_title, description, address\_id, office\_id)

VALUES ( 'Frontend Developer', 'Frontend developer for developing demanding webpages',1 ,2),

('IT technician', 'IT technician to maintain the computer networks and providing tech support', 2, 1),

('R&D Leader', 'R&D Leader to steer a engineer team to develop new products', 2, 3);

INSERT INTO public.tbl\_faculty(

name, org\_id)

VALUES ( 'Engineering',1),

('Science',1),

('Fine Arts',2),

('Pharmacy',2);

INSERT INTO public.tbl\_department(

name, faculty\_id, address\_id)

VALUES ( 'Computer Science' ,1 ,3),

('Biochemistry' ,2 ,4),

('Statue' ,3 ,3),

('Pharmaseutical Technologies' ,4 ,4);

INSERT INTO public.tbl\_graduate\_level(

name, dept\_id)

VALUES ( 'Bachelors Degree' ,1),

('Associate Degree' ,2),

('Masters Degree' ,3),

('Doctoral degree' ,4);

CALL add\_person\_student('Deniz', 'Yurekdeler', '05544413061', 'dyurekdeler@gmail.com' , '1997-07-03', 1, 3.6, '4')

CALL add\_person\_student('Cem', 'Corbacioglu', '05544413061', 'cemcorbacioglu@gmail.com', '1995-01-06', 2, 3.5 , '4')

CALL add\_person\_student('Aybars', 'Kokce' , '05544413061', 'akokce@hotmail.com', '1993-11-09', 3, 1.9, '3' )

CALL add\_person\_member('Ege', 'Kubilay', '05544413061', 'ekubilay@gmail.com', '1990-02-05', 3, '12345678')

CALL add\_person\_member('Melisa', 'Erdem' , '05544413061', 'merdem@aol.com', '1985-01-02', 2, '12345678')

CALL add\_person\_member('Hakan', 'Atilgan', '05544413061' , 'hatilgan@hotmail.com' , '1992-11-10', 3,'12345678')

CALL add\_person\_teacher('Ahmet', 'Atakus', '05544413061', 'ahmetatakus@hotmail.com', '1980-05-07', 3, 'Linguistic', 1)

CALL add\_person\_teacher('Bilge', 'Usak' , '05544413061', 'bilgeusak@hotmail.com' , '1994-10-11', 3, 'Object Oriented' ,2)

INSERT INTO public.tbl\_course(

name, code, credit, dep\_id)

VALUES ( 'Algorithms', 'BIL115115', 7, 1),

( 'Carving', 'STA223344', 6, 3),

( 'Organic modeling', 'BIO11345', 5, 2);

INSERT INTO public.tbl\_project(

title, description, course\_id)

VALUES ( 'Sorting algorithm project', 'describe each algorithms characteristics', 1),

( 'Organical modeling examples project', 'draw 10 organical compunds model', 3);

INSERT INTO public.tbl\_educates\_in(

start\_date, end\_date, gpa, grad\_id, person\_id)

VALUES ('2013-11-07', '2017-07-07', 2.9, 1, 1),

('2015-11-05', '2017-07-07', 2.9, 4, 2),

('2016-11-03', '2018-08-08', 2.9, 3, 3);

INSERT INTO public.tbl\_references(

teacher\_id, referenced\_person\_id, date, context)

VALUES (4, 1, '2019-12-12', 'Bright kid'),

(4, 2, '2020-01-01', 'Best in her class');

INSERT INTO tbl\_messages(from\_person\_id, to\_person\_id, title, context, date) VALUES

(1,3,’Sınav Saati Hk.’, ‘Sınav saati ne zaman?’,’01.01.2020’),

(4,2,’Ders Programı’, ‘Ders programında Salı günü hangi ders var?’);

INSERT INTO public.tbl\_knows(

person\_id, skill\_id)

VALUES (1, 3),

(2, 3),

(4, 2);

INSERT INTO public.tbl\_teaches(

teacher\_id, course\_id, semester)

VALUES (4, 1, 'winter'),

(5, 2,'winter'),

(3, 3, 'spring');

INSERT INTO public.tbl\_works\_on(

student\_id, project\_id, grade, semester)

VALUES (1, 1, '40' ,'spring'),

(2, 1, '50', 'spring');

INSERT INTO public.tbl\_enrolls(

student\_id, course\_id, semester, grade)

VALUES (1, 1, 'spring', 1.9),

(2, 1, 'spring', 1.9);

1. TRIGGERS SCRIPT

CREATE OR REPLACE FUNCTION add\_active\_student()

RETURNS trigger AS

$$

BEGIN

IF NEW.end\_date = Null AND assert\_is\_student = 'False' THEN

INSERT INTO tbl\_student(student\_id,gpa,grade)

VALUES(NEW.person\_id,NEW.gpa,'1');

END IF;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER add\_active\_student\_trigger

AFTER INSERT

ON tbl\_educates\_in

FOR EACH ROW

EXECUTE PROCEDURE add\_active\_student();

CREATE OR REPLACE FUNCTION init\_employment()

RETURNS trigger AS

$$

BEGIN

IF NEW.status = 'ACCEPTED' THEN

WITH new\_worksfor as (

SELECT job\_title, member\_id, office\_id FROM tbl\_job\_offer , NEW WHERE offer\_id = NEW.offer\_id )

INSERT INTO tbl\_works\_for(member\_id, job\_title, office\_id, start\_date)

SELECT member\_id,job\_title, office\_id FROM new\_worksfor, current.date;

END IF;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER init\_employment\_trigger

AFTER INSERT OR UPDATE

ON tbl\_applies

FOR EACH ROW

EXECUTE PROCEDURE init\_employment();

CREATE OR REPLACE FUNCTION remove\_student()

RETURNS trigger AS

$$

BEGIN

IF NEW.end\_date != Null THEN

DELETE FROM tbl\_student WHERE student\_id = OLD.student\_id;

END IF;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER remove\_student\_trigger

AFTER UPDATE

ON tbl\_educates\_in

FOR EACH ROW

EXECUTE PROCEDURE remove\_student();

CREATE OR REPLACE FUNCTION auto\_join\_group()

RETURNS trigger AS

$$

BEGIN

INSERT INTO tbl\_joins(member\_id, group\_id) VALUES ( NEW.member\_id, NEW.group\_id);

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER auto\_join\_group\_trigger

AFTER INSERT

ON tbl\_group

FOR EACH ROW

EXECUTE PROCEDURE auto\_join\_group();

1. SELECTION SCRIPTS

A)

INSERT INTO tbl\_skill(name,category) VALUES ('Android Developement','Software');

UPDATE tbl\_skill SET name='Android Mobile Development' WHERE skill\_id = 4;

DELETE FROM tbl\_skill WHERE skill\_id = 4;

INSERT INTO tbl\_job\_offer VALUES (job\_title, description, address\_id , office\_id) VALUES ('Müzik Öğretmeni', 'Ege Üniversitesinde 1. sınıflara müzik eğitimi verecek öğretmen', 1, 1);

UPDATE tbl\_job\_offer SET job\_title='Halk Müziği Öğretmeni', description='Ege üniversitesinde ortak seçmeli derste halk müziği dersi verecek öğretmen', address\_id=1, office\_id=1 WHERE offer\_id = 5;

DELETE FROM tbl\_job\_offer WHERE offer\_id = 5;

INSERT INTO tbl\_works\_for(member\_id, start\_date, job\_title, office\_id) VALUES (2, '12.12.2016', 'RD Engineer' , 1);

UPDATE tbl\_works\_for SET , end\_date='05.09.2019', job\_title='Senior RD Engineer'

WHERE member\_id = 2, job\_title = 'RD Engineer', office\_id=1;

DELETE FROM tbl\_works\_for WHERE member\_id = 2, job\_title = 'Senior RD Engineer', office\_id=1;

B)

i)

SELECT name, description FROM tbl\_group

SELECT branch FROM tbl\_teacher

SELECT name FROM tbl\_faculty

SELECT fname , lname, bday FROM tbl\_person

WHERE bday >

(SELECT bday FROM tbl\_personWHERE fname='Ege' AND lname='Kubilay')

ii)

SELECT name

FROM tbl\_course C

INNER JOIN tbl\_enrolls E ON C.course\_id = E.course\_id

WHERE student\_id = 1;

SELECT fname

FROM tbl\_person

WHERE address\_id =

(SELECT address\_id

FROM tbl\_address

WHERE Country='Turkey' AND (City='Ankara' OR City='Bursa'))

SELECT gpa, fname, lname

FROM tbl\_person

INNER JOIN tbl\_student ON person\_id = student\_id

SELECT country, city ,street, name

FROM tbl\_department D

INNER JOIN tbl\_address A ON D.address\_id = A.address\_id

SELECT D.name , C.name

FROM tbl\_course C

INNER JOIN tbl\_department D ON dep\_id = dept\_id

iii)

SELECT grade

FROM tbl\_project P

INNER JOIN tbl\_course C ON P.course\_id = C.course\_id

INNER JOIN tbl\_enrolls E ON P.course\_id = E.course\_id

WHERE student\_id =1

SELECT fname, lname , W.grade

FROM tbl\_person P

INNER JOIN tbl\_student ON student\_id = person\_id

INNER JOIN tbl\_works\_on W ON W.student\_id = P.person\_id

SELECT fname, lname, context

FROM tbl\_person

INNER JOIN tbl\_references

ON referenced\_person\_id =person\_id

INNER JOIN tbl\_student ON student\_id = person\_id

WHERE gpa>3.0

SELECT fname, lname

FROM tbl\_person

WHERE EXISTS(

SELECT \* FROM tbl\_educates\_in EI

INNER JOIN tbl\_person P ON EI.person\_id = P.person\_id

INNER JOIN tbl\_graduate\_level G ON EI.grad\_id = G.grad\_id

WHERE name= 'Associate Degree' )

C)

SELECT fname,lname

FROM tbl\_person

WHERE person\_id=

(SELECT teacher\_id FROM tbl\_teacher

WHERE dept\_id= (SELECT dept\_id

FROM tbl\_department WHERE name = 'Computer Science' ))

SELECT fname, lname ,E.grade

FROM tbl\_enrolls E

INNER JOIN tbl\_student S ON E.student\_id = S.student\_id

INNER JOIN tbl\_person ON person\_id = S.student\_id

INNER JOIN tbl\_course C on E.course\_id = C.course\_id

INNER JOIN tbl\_teaches T ON T.course\_id = C.course\_id

WHERE teacher\_id=7 AND E.semester='winter 2019-2020' AND E.semester = T.semester;

SELECT fname, lname FROM tbl\_person

WHERE person\_id IN

(SELECT student\_id FROM tbl\_student

UNION

SELECT member\_id FROM tbl\_member)

SELECT fname, lname, branch, name

FROM tbl\_teacher T

INNER JOIN tbl\_person ON teacher\_id = person\_id

INNER JOIN tbl\_department D ON T.dept\_id = D.dept\_id

SELECT C.name

FROM tbl\_course C

INNER JOIN tbl\_department D ON C.dep\_id = D.dept\_id

INNER JOIN tbl\_teaches T ON T.course\_id = C.course\_id

WHERE teacher\_id = 7 AND semester='winter 2019-2020';