Final Defense Presentation

STUDENT

Duman Yessenbay
Zhassur Tashpulat

LECTURER

Maxat Skakov

Description

Database topic - University

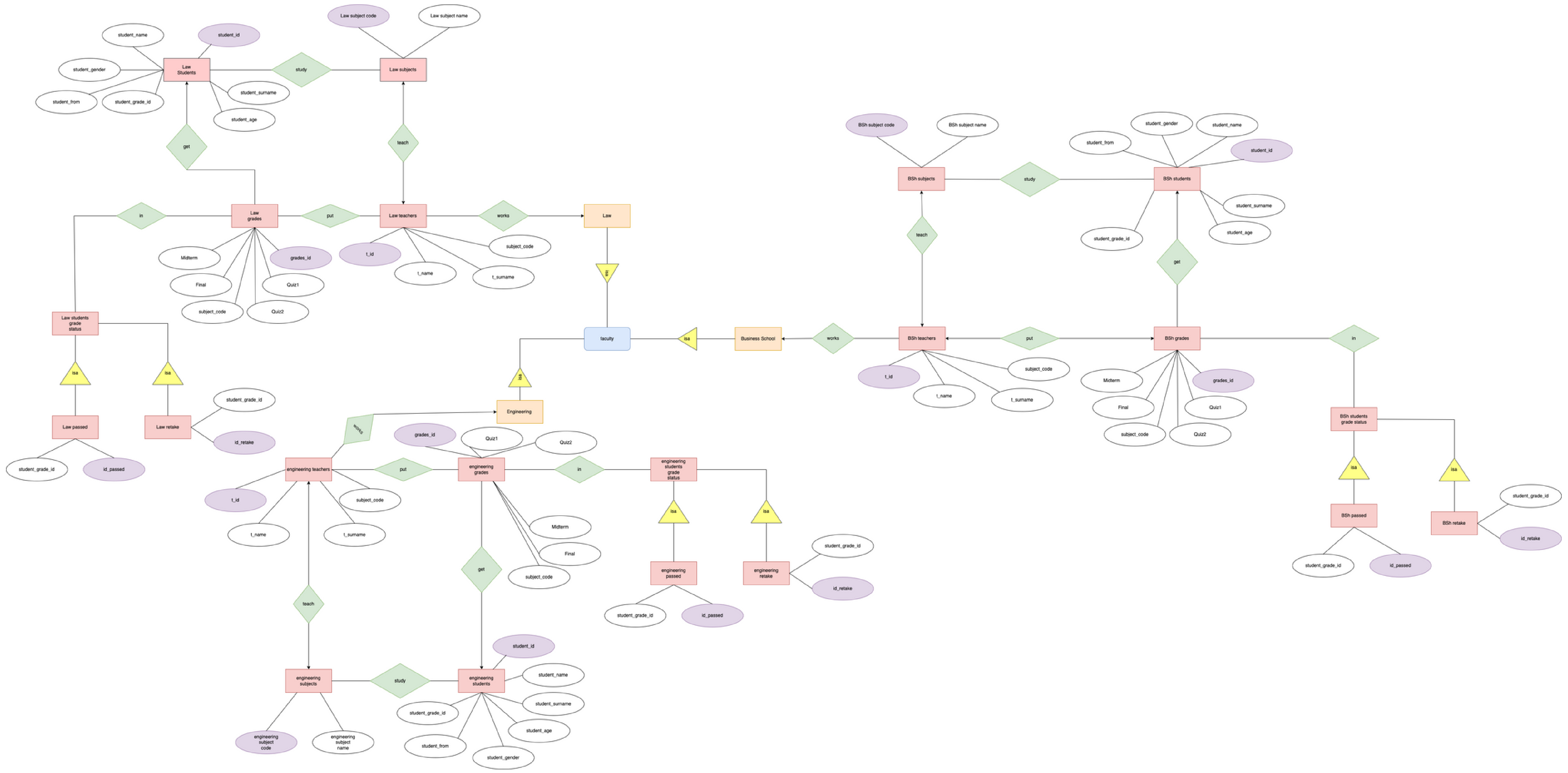
Our database will be used by teachers and university students

The database is designed for grading and getting grades

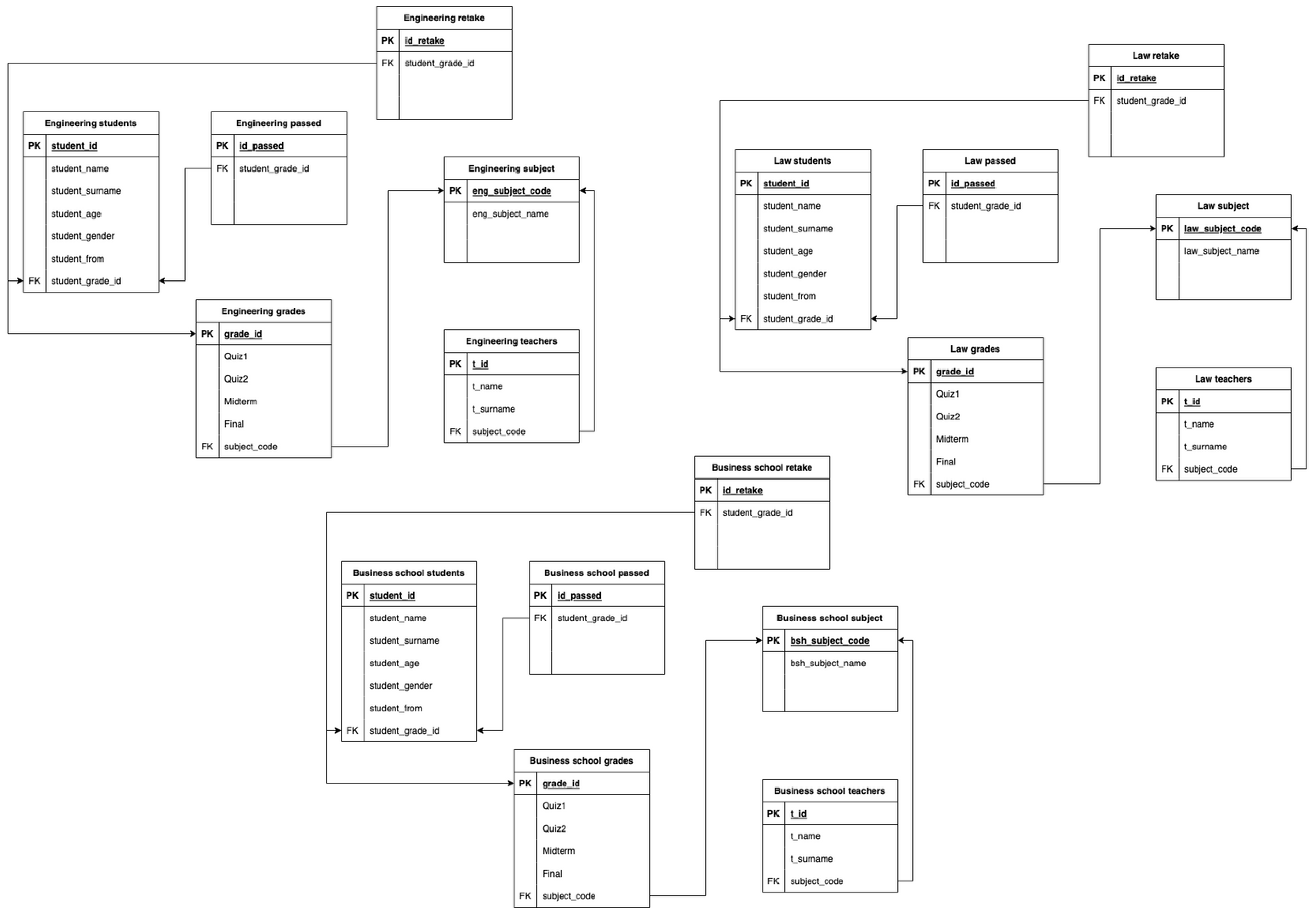
The end users of the database are teachers and students

We took the idea from real life

ER diagram


Law subject code
Law subject name
student_name
student_id
student_gender
Law Students
study
Law subjects
student_from
student_grade_id
student_surname
student_age
get
teach
in
Law grades
put
Law teachers
works
Law
Midterm
grades_id
t_id
subject_code
Final
Quiz1
t_name
t_surname
subject_code
Quiz2
isa
Law students grade status
faculty
isa
Business School
works
BSh teachers
put
BSh grades
in
isa
isa
isa
Engineering
Law passed
Law retake
student_grade_id
id_retake
student_grade_id
id_passed
works
grades_id
Quiz1
Quiz2
engineering teachers
put
engineering grades
in
engineering students grade status
t_id
subject_code
Midterm
isa
isa
t_name
t_surname
Final
get
subject_code
engineering passed
engineering retake
student_grade_id
id_retake
student_grade_id
id_passed
teach
student_id
engineering subjects
study
engineering students
student_name
student_surname
student_grade_id
student_age
student_from
student_gender
engineering subject code
engineering subject name
BSh subject code
BSh subject name
student_gender
student_name
student_from
student_id
BSh subjects
study
BSh students
student_surname
student_grade_id
student_age
teach
get
t_id
subject_code
t_name
t_surname
Midterm
grades_id
Final
Quiz1
subject_code
Quiz2
BSh students grade status
isa
isa
BSh passed
BSh retake
student_grade_id
id_retake
student_grade_id
id_passed

Relational schema



Functional Dependencies

Engineering_students: student_id-->{student_name, student_surname, student_age, student_gender, student_from, student_grade_id}

Engineering teachers: t_id-->{t_name, t_surname, subject_code}

Engineering subject: subject_code-->{subject_name}

Engineering grades: grade_id-->{Quiz1, Quiz2, Midterm, Final, subject_code}

Engineering passed: id_passed-->{student_grade_id}

Engineering_retake: id_retake-->{student_grade_id}

Law_students: student_id-->{student_name, student_surname, student_age, student_gender, student_from, student_grade_id}

Law teachers: t_id-->{t_name, t_surname, subject_code}

Law subject: subject_code-->{subject_name}

Law grades: grade_id-->{Quiz1, Quiz2, Midterm, Final, subject_code}

Law passed: id_passed-->{student_grade_id}

Law_retake: id_retake-->{student_grade_id}

BSH_students: student_id-->{student_name, student_surname, student_age, student_gender, student_from, student_grade_id}

BSH teachers: t_id-->{t_name, t_surname, subject_code}

BSH subject: subject_code-->{subject_name}

BSH grades: grade_id-->{Quiz1, Quiz2, Midterm, Final, subject_code}

BSH passed: id_passed-->{student_grade_id}

BSH_retake: id_retake-->{student_grade_id}

IV

Queries

1.selection) SELECT * FROM engineering_students WHERE student_name LIKE 'A%';
σ[student_name like 'A%'](engineering_students)

2.projection) SELECT student_id, student_name, student_from FROM `engineering_students` where student_id =20;
π[student_id, student_name_student_from](σ[student_id](engineering_students)

3.update) update engineering_grades SET Quiz1 = 7 WHERE grade_id = 1;
engineering_grades <-- π[Quiz1](σ[grade_id=1](engineering_grades))

4.alter add) ALTER TABLE engineering_grades ADD COLUMN total int

5.rename) select total as total_points from `engineering_grades` ;
ρ[total_grades](π[total](engineering_grades)

6.delete) DELETE FROM engineering_students WHERE student_id = 122;
engineering_students <-- engineering_students - σ[student_id=122](engineering_students)

7.order by) SELECT (Quiz1+Quiz2+Midterm+Final) as total FROM engineering_grades ORDER BY total;
τ[total](σ[Quiz1+Quiz2+Midterm+Final](engineering_grades));

8.group by) SELECT student_gender FROM engineering_students GROUP BY student_gender;
ɣ[student_gender](π[student_gender](engineering_students));

IV Queries

9.join) SELECT engineering_teacher.t_name, engineering_subject.eng_subject_name FROM engineering_teacher JOIN engineering_subject ON
engineering_teacher.subject_code = engineering_subject.eng_subject_code;
π[engineering_teachers.t_name, engineering_subject.engineering_subject_name](engineering_teacher)
engineering_teacher ⋈ engineering_subject [subject_code = subject_code]

10.subquery) SELECT student_name, student_age FROM engineering_students
WHERE student_age = ANY (SELECT student_age FROM engineering_students WHERE student_age > 20);
Step 1: π[σ[student_age>20](engineering_students) as sq](engineering_students)
Step 2: π[student_name, student_age](σ[student_age = sq);

11.aggregation) SELECT MAX(Quiz1+Quiz2+Midterm+Final) as totalMax FROM engineering_grades;
ɣ[MAX(Quiz1+Quiz2+Midterm+Final) as totalmax](engineering_grades

12.union)
SELECT engineering_students.student_name FROM engineering_students
UNION
SELECT law_students.student_name FROM law_students;
UNION
SELECT bsh_students.student_name FROM bsh_students

π[student_name](engineering_students) U π[student_name](law_students) U π[student_name](bsh_students)

Queries

13.intersection)

SELECT DISTINCT engineering_students.student_name FROM engineering_students INNER JOIN law_students ON engineering_students.student_name = law_students.student_name INNER JOIN bsh_students ON engineering_students.student_name = bsh_students.student_name;

Dublicate elimination[π[student_name](engineering_students)
engineering_students inner⋈ law_students. [student_name = student_name]
engineering_students inner⋈ bsh_students. [student_name = student_name]](engineering_students)

14.except)

SELECT student_age from engineering_students
WHERE student_age not in (SELECT student_age FROM engineering_students WHERE student_age<20);

Step 1: π[σ[student_age<20](engineering_students) as step1](engineering_students)
Step 2: π[π[student_age](engineering_students) as step2](engineering_students)
Step 3: σ[student_age not in step1](student_age)

15.null)

SELECT * FROM `engineering_grades` WHERE final is null;
σ[final is null](engineering_grades)

Queries

16.views)
CREATE VIEW MaxAvgMinTotals as
SELECT MAX(Quiz1+Quiz2+Midterm+Final) as maxTotal, AVG(Quiz1+Quiz2+Midterm+Final) as avgTotal, MIN(Quiz1+Quiz2+Midterm+Final) as minTotal from engineering_grades;

create view maxavgmintotals as <
step 1: π[ɣ[Quiz1+Quiz2+Midterm+Final)as maxtotal] as maxTotal](engineering_grades)
step 2: π[ɣ[Quiz1+Quiz2+Midterm+Final)as avgtotal] as avgTotal](engineering_grades)
step 3: π[ɣ[Quiz1+Quiz2+Midterm+Final)as mintotal] as minTotal](engineering_grades)
step 4: π[maxTotal, avgTotal, minTotal](engineering_grades) >;

17.constraints)
CREATE INDEX studentID_Name on engineering_students(student_id, student_name);

IV

Queries

18) before update trigger

```
DROP TRIGGER IF EXISTS `update_grade_id_in_EnginGrades`;CREATE DEFINER=`root`@`localhost` TRIGGER `update_grade_id_in_EnginGrades`
BEFORE UPDATE ON `engineering_students` FOR EACH ROW BEGIN
UPDATE engineering_grades SET engineering_grades.grade_id = new.student_grade_id
where engineering_grades.grade_id = old.student_grade_id;
END
```

19) before insert trigger

```
DROP TRIGGER IF EXISTS `insert_grade_id_to_EnginGrades`;CREATE DEFINER=`root`@`localhost` TRIGGER `insert_grade_id_to_EnginGrades`
BEFORE INSERT ON `engineering_students` FOR EACH ROW BEGIN
INSERT INTO engineering_grades(engineering_grades.grade_id)
VALUES(new.student_grade_id);
END
```

20)after delete trigger

```
DROP TRIGGER IF EXISTS `delete_grade_id_from_EnginGrades`;CREATE DEFINER=`root`@`localhost` TRIGGER
`delete_grade_id_from_EnginGrades` AFTER DELETE ON `engineering_students` FOR EACH ROW BEGIN
DELETE FROM engineering_grades WHERE
engineering_grades.grade_id = old.student_grade_id;
END
```

V

Transaction and Index

Transaction:

```
START TRANSACTION;
UPDATE bsh_grades set Final = Final+5 WHERE grade_id=1;
UPDATE bsh_grades set Final = Final-5 WHERE grade_id=2;
COMMIT;
```

Index:

```
CREATE INDEX studentID_Name on law_students(student_id, student_name);
```