LIMKOKWING UNIVERSITY OF CREATIVE TECHNOLOGY

CAMBODIA

BIE2153 • DATABASE SYSTEMS

GROUP ASSIGNMENT

Academic Honesty Policy Statement

I, hereby attest that the contents of this attachment are my own work. Referenced works, articles, art, programs, papers or parts thereof are acknowledged at the end of this paper. This includes data excerpted from CD-ROMs, the Internet, other private networks, and other people's disk of the computer system.

Submitted by

Name	: Hing Zodiac Jack Rearsice
ID	: 601005839
Signature	: [illegible]

Under the supervision of
Lecturer LENG RATHANA

LECTURER'S COMMENTS/GRADES	*For office use only*
	Date:
	Time:
	RECEIVER'S NAME: LENG RATHANA

TABLE OF CONTENTS

Chapter 1:

Introduction to the Business Entity:

An online exam system that provides a platform for creating, managing, and evaluating exams. This system will cater to educational institutions, training centers, and organizations, allowing them to conduct online assessments efficiently and securely. The database system will store user information, exam details, questions, and individual student results.

Overview of the Proposed Database System:

The proposed database system for the online exam platform will consist of several key entities:

Course: Represents a course offered by an educational institution. It contains information such as the courseID and course name.

Student: Represents a student enrolled in the educational institution. It includes details such as student ID, name, email, and personal information.

Enroll: Represents the enrollment of a student in a particular course. It associates a student with a course and may include additional details like enrollment date.

Instructor: Represents an instructor or teacher who teaches courses. It includes attributes like instructor ID, name, email, and personal information.

Exam: Represents an exam or assessment associated with a specific course. It includes details such as exam ID, title, duration, and date.

Participation: Represents the participation of a student in an exam. It associates a student with an exam and may include additional details like start time and end time.

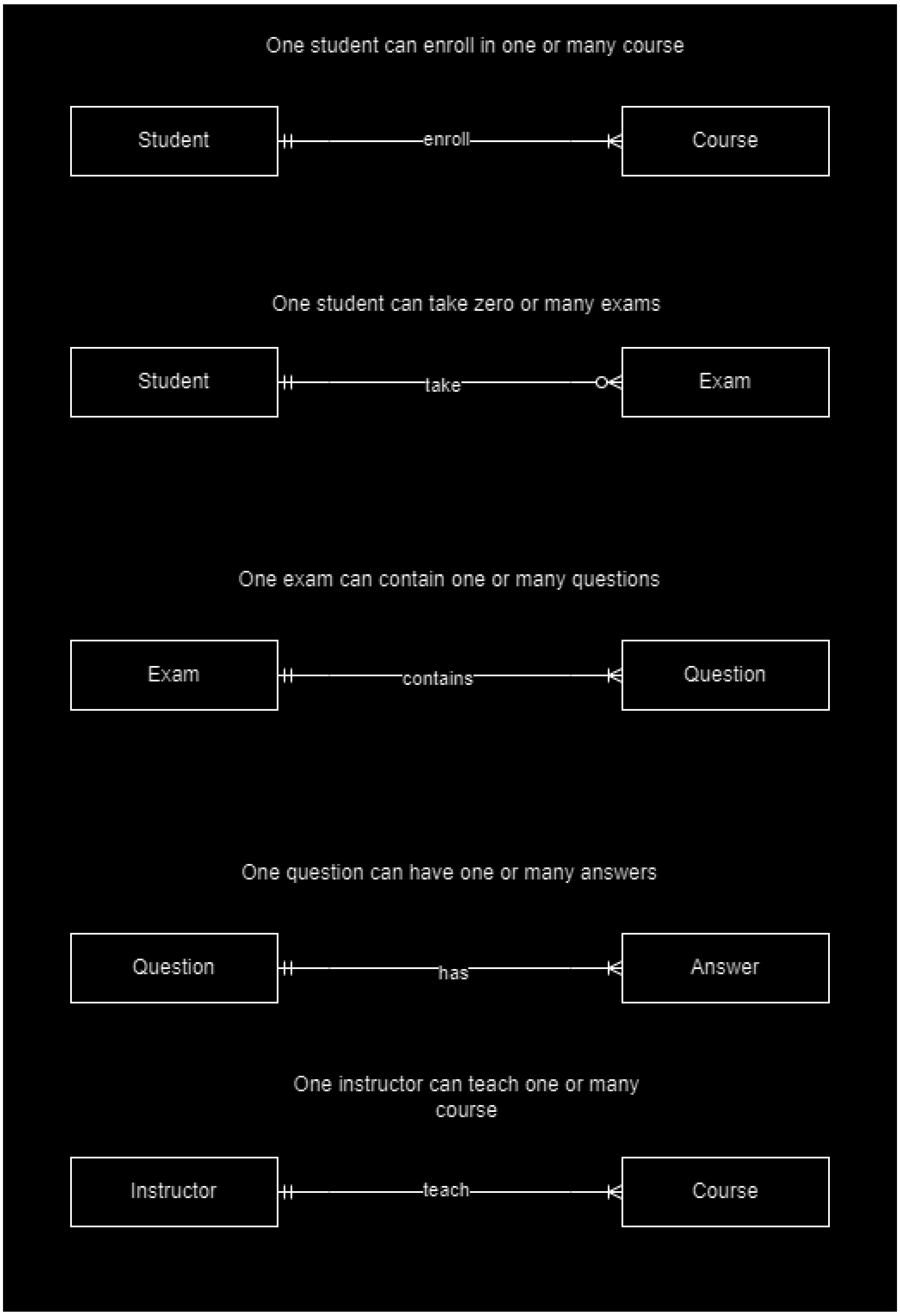
Question: Represents a question or problem in an exam. It includes attributes such as question ID and its content.

Result: Represents the result or score of a student in an exam. It associates a student with an exam and includes attributes like score and grade.

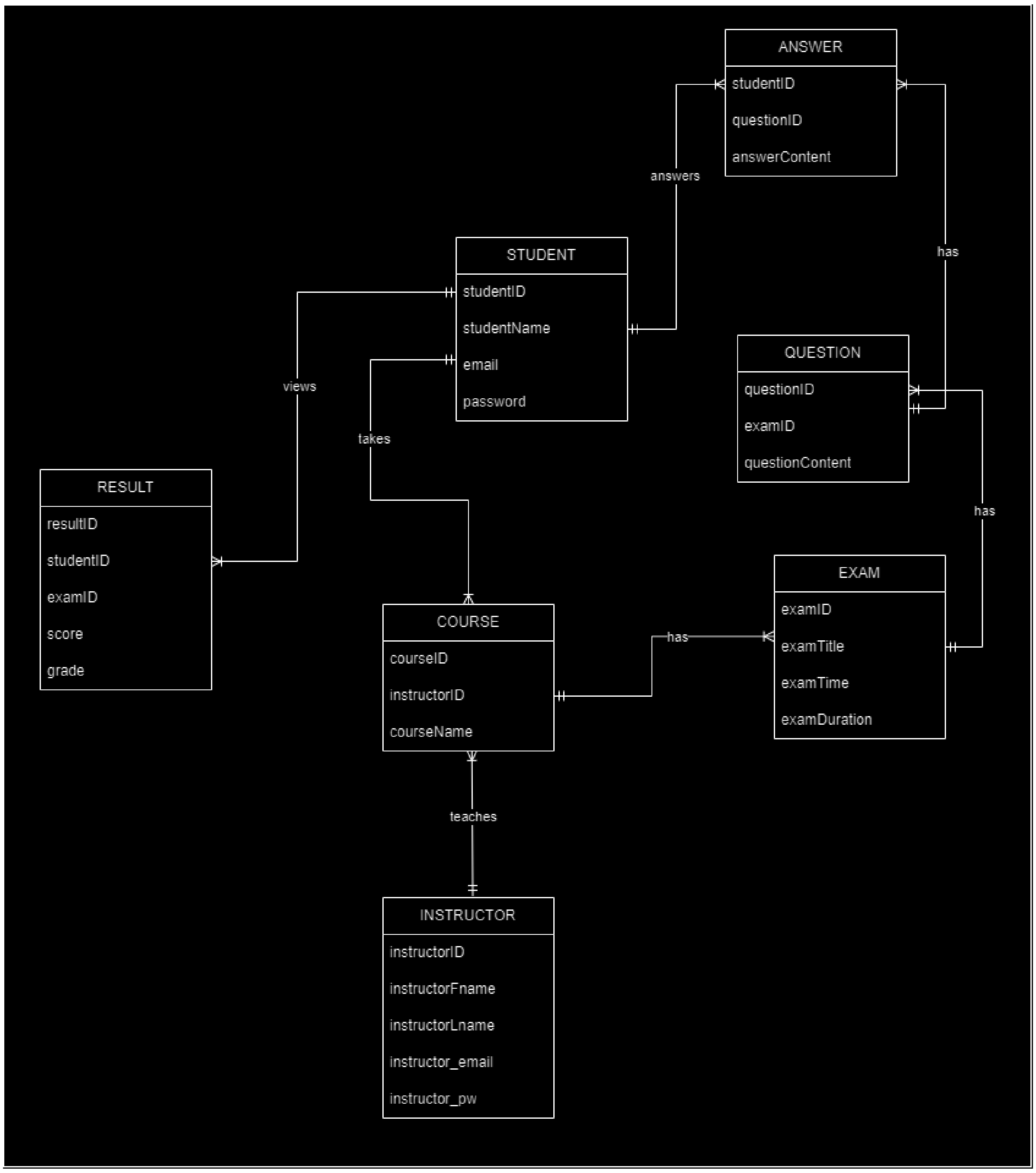
Chapter 2:

Entity Relationship Diagram (ERD):

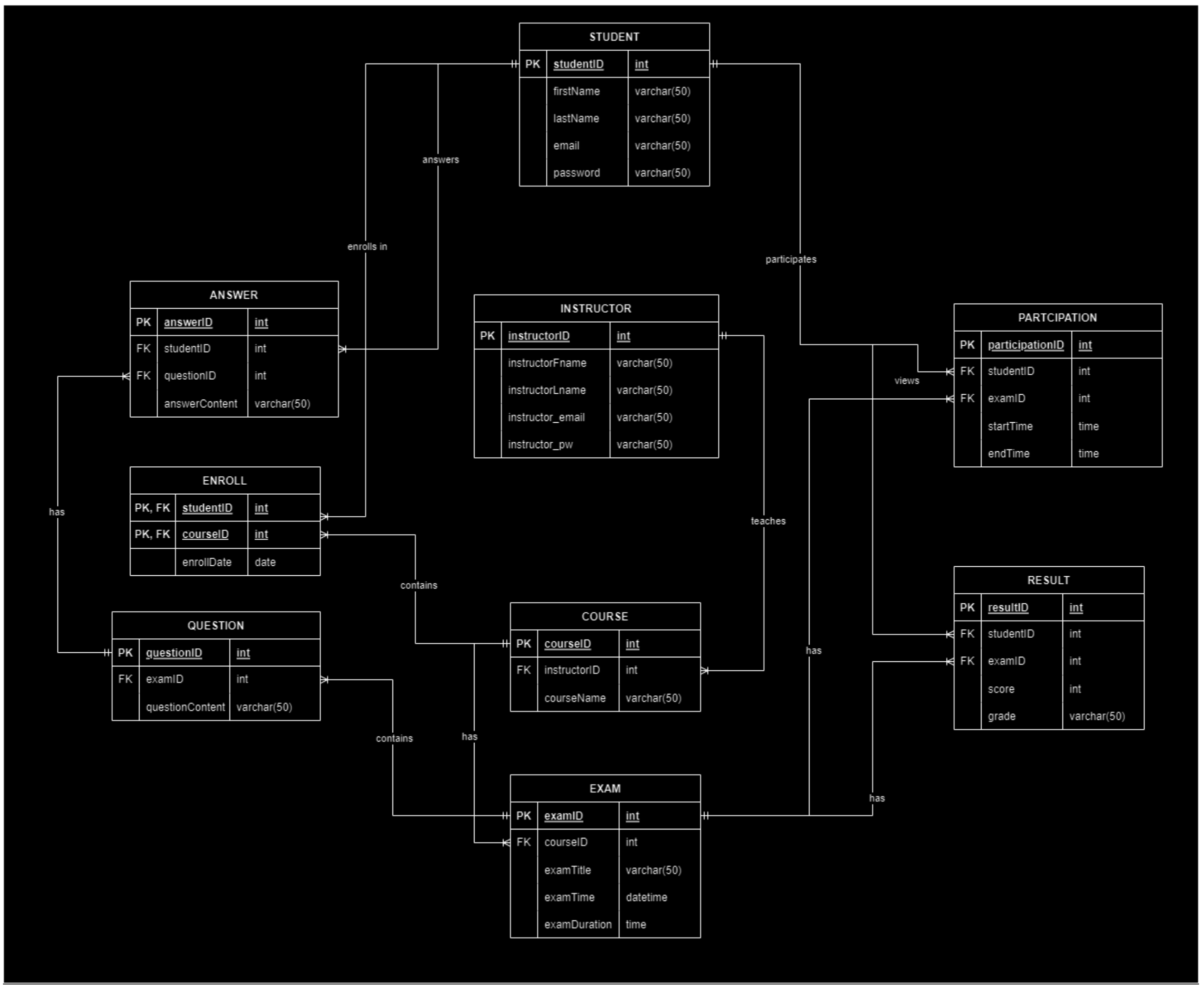
> Conceptual ERD:


One student can enroll in one or many course
Student
enroll
Course
One student can take zero or many exams
Student
take
Exam
One exam can contain one or many questions
Exam
contains
Question
One question can have one or many answers
Question
has
Answer
One instructor can teach one or many course
Instructor
teach
Course

> Logical ERD:



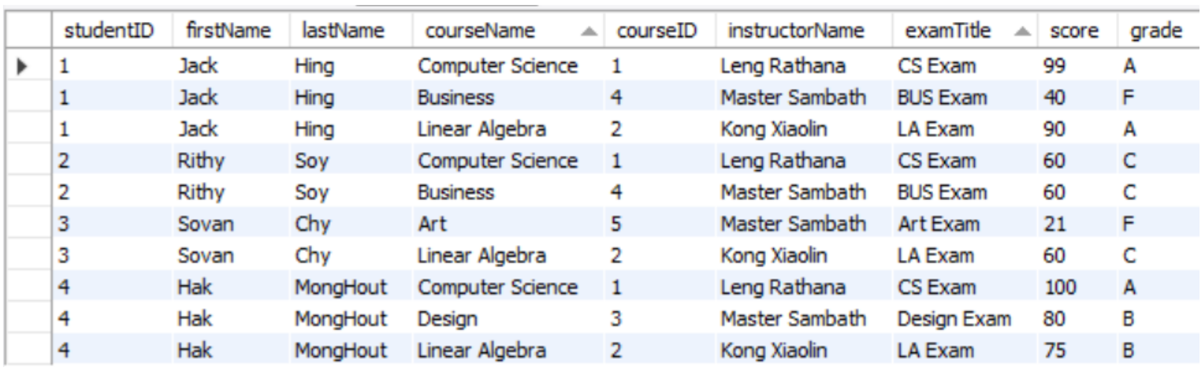
> Physical ERD:



Normalization Table:

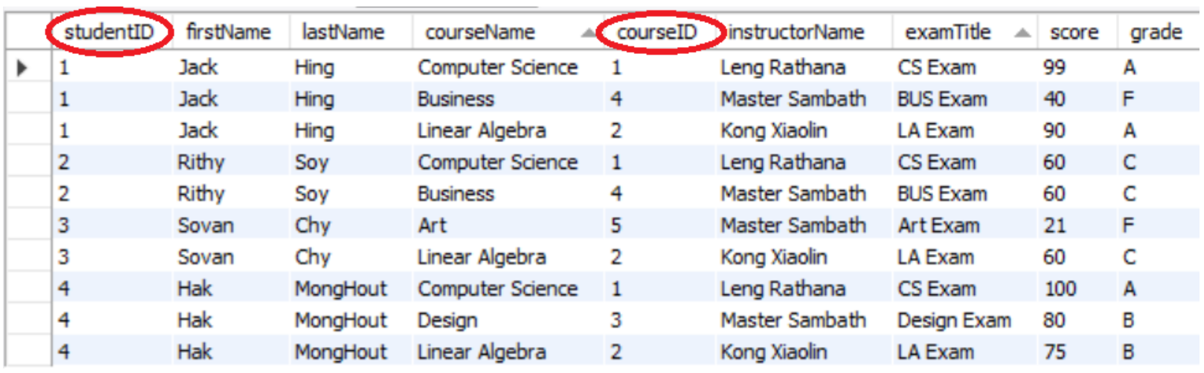
First normalization

+ No null values
+ Single value attributes
+ Unique name for every column



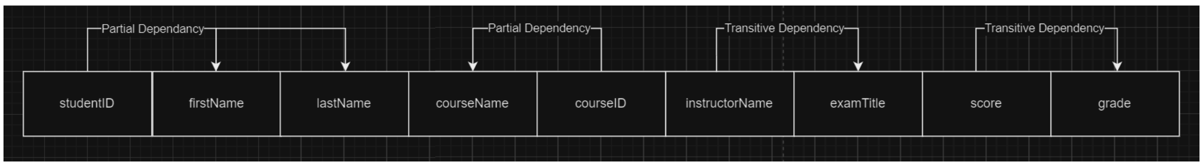
studentID	firstName	lastName	courseName	courseID	instructorName	examTitle	score	grade
1	Jack	Hing	Computer Science	1	Leng Rathana	CS Exam	99	A
1	Jack	Hing	Business	4	Master Sambath	BUS Exam	40	F
1	Jack	Hing	Linear Algebra	2	Kong Xiaolin	LA Exam	90	A
2	Rithy	Soy	Computer Science	1	Leng Rathana	CS Exam	60	C
2	Rithy	Soy	Business	4	Master Sambath	BUS Exam	60	C
3	Sovan	Chy	Art	5	Master Sambath	Art Exam	21	F
3	Sovan	Chy	Linear Algebra	2	Kong Xiaolin	LA Exam	60	C
4	Hak	MongHout	Computer Science	1	Leng Rathana	CS Exam	100	A
4	Hak	MongHout	Design	3	Master Sambath	Design Exam	80	B
4	Hak	MongHout	Linear Algebra	2	Kong Xiaolin	LA Exam	75	B

+ Identify primary key



studentID	firstName	lastName	courseName	courseID	instructorName	examTitle	score	grade
1	Jack	Hing	Computer Science	1	Leng Rathana	CS Exam	99	A
1	Jack	Hing	Business	4	Master Sambath	BUS Exam	40	F
1	Jack	Hing	Linear Algebra	2	Kong Xiaolin	LA Exam	90	A
2	Rithy	Soy	Computer Science	1	Leng Rathana	CS Exam	60	C
2	Rithy	Soy	Business	4	Master Sambath	BUS Exam	60	C
3	Sovan	Chy	Art	5	Master Sambath	Art Exam	21	F
3	Sovan	Chy	Linear Algebra	2	Kong Xiaolin	LA Exam	60	C
4	Hak	MongHout	Computer Science	1	Leng Rathana	CS Exam	100	A
4	Hak	MongHout	Design	3	Master Sambath	Design Exam	80	B
4	Hak	MongHout	Linear Algebra	2	Kong Xiaolin	LA Exam	75	B

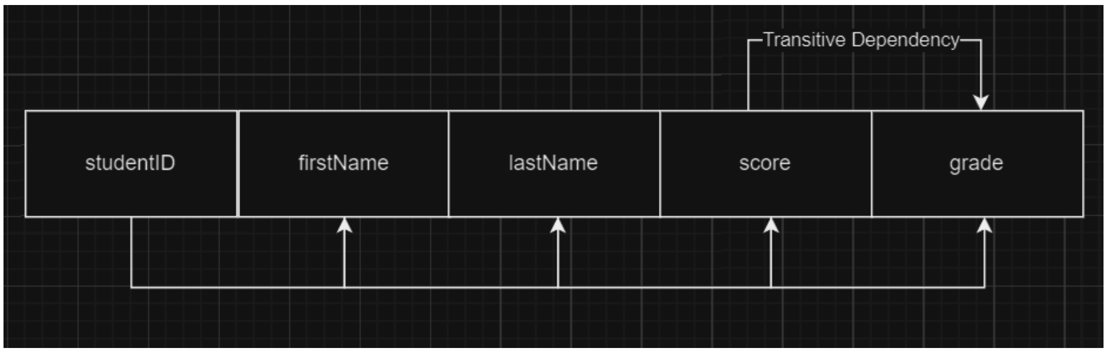
- **Identify Dependencies**



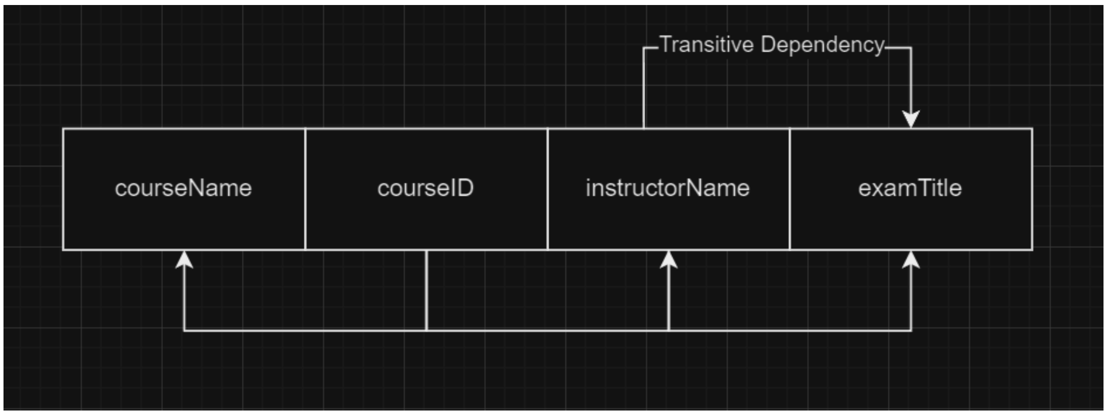
Second Normalization

- **Identify Tables to remove partial dependency**

-> Student Table:

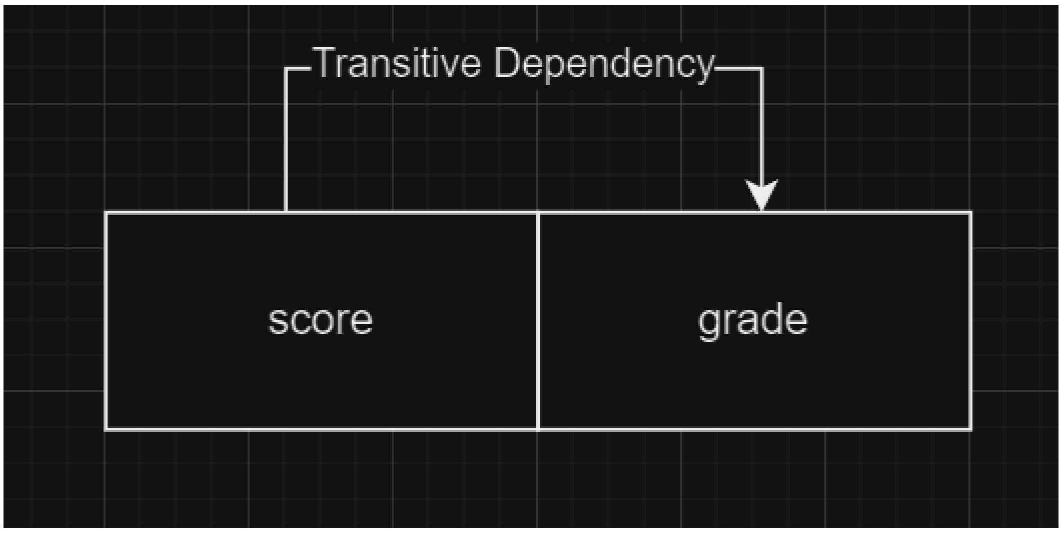


-> Course Table:

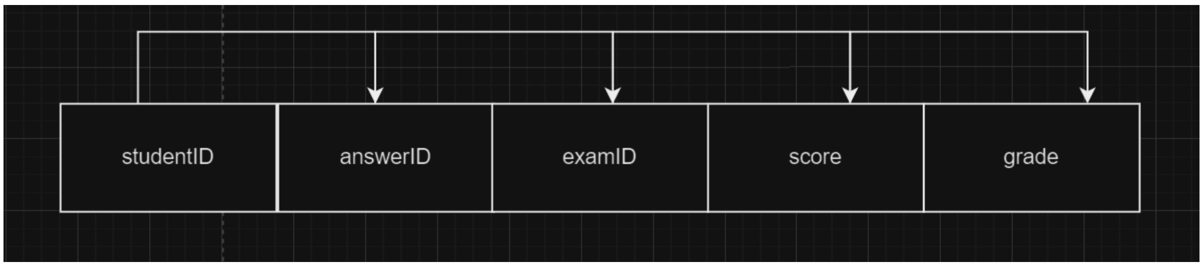


Third Normalization

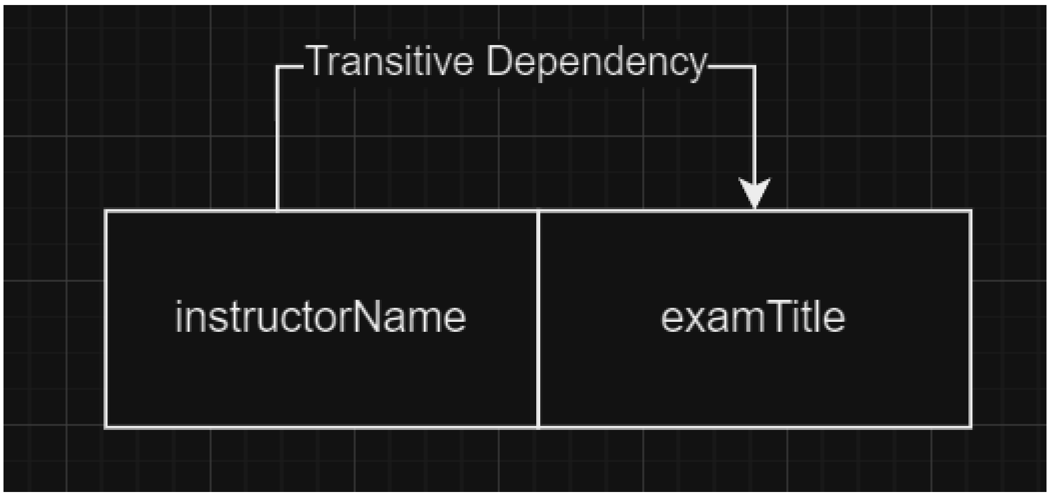
- **Identify Tables to remove transitive dependency**



Used to form another table called Result table:

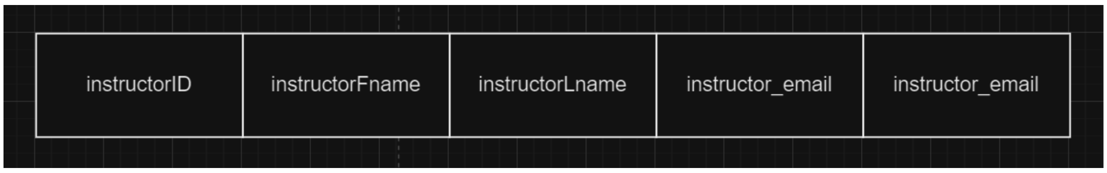


Then



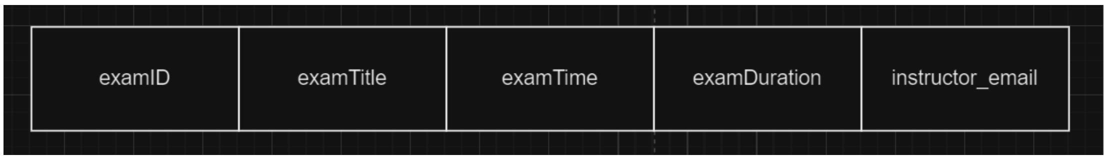
Used to form 2 other tables to eliminate the transitive dependency:

-> Instructor Table:



instructorID	instructorFname	instructorLname	instructor_email	instructor_email

-> Exam Table:



examID	examTitle	examTime	examDuration	instructor_email

Data Dictionary:

Table Name	Attribute Name	Type	Range	PK or FK	Required
Student	studentID	int	0-N	PK	yes
	firstName	varchar(50)	0-50		no
	lastName	varchar(50)	0-50		no
	email	varchar(50)	0-50		no
	password	varchar(50)	0-50		no
Answer	answerID	int	0-N	PK	yes
	questionID	int	0-N	FK	no
	studentID	int	0-N	FK	no
	answerContent	varchar(50)	0-50		no
Course	courseID	int	0-N	PK	yes
	instructorID	int	0-N	FK	no
	courseName	varchar(50)	0-50		no
Enroll	studentID	int	0-N	PK, FK1	yes
	courseID	int	0-N	PK, FK2	yes
	enrollDate	date	0-N		no
Exam	examID	int	0-N	PK	yes
	courseID	int	0-N	FK	no
	examTitle	varchar(50)	0-50		no
	examTime	datetime	0-N		no
	examDuration	time	0-N		no
Instructor	instructorID	int	0-N	PK	yes
	instructorFname	varchar(50)	0-50		no
	instructorLname	varchar(50)	0-50		no
	instructorEmail	varchar(50)	0-50		no
	instructor_pw	varchar(50)	0-50		no

Participation	participationID	int	0-N	PK	yes
	studentID	int	0-N	FK	no
	examID	int	0-N	FK	no
	startTime	time	0-N		no
	endTime	time	0-N		no
Question	questionID	int	0-N	PK	yes
	examID	int	0-N	FK	no
	questionContent	varchar(50)	0-50		no
Result	answerID	int	0-N	PK, FK1	yes
	studentID	int	0-N	PK, FK2	yes
	examID	int	0-N	PK, FK3	yes
	score	int	0-N		no
	grade	varchar(50)	0-50		no

Chapter 3:

Implementation and loading

Database Creation

```
create database online_exam_system;
use online_exam_system;
create table student(
 studentID int primary key not null,
 firstName varchar(50),
  lastName varchar(50),
  email varchar(50),
  password varchar(50)
);
create table instructor(
 instructorID int primary key not null,
 instructorFname varchar(50),
  instructorLname varchar(50),
  instructor_email varchar(50),
  instructor_pw varchar(50)
);
create table participation (
 participationID int primary key not null,
  studentID int,
  examID int,
  startTime time,
  endTime time
);
```

```
create table course (
  courseID int primary key not null,
  instructorID int,
  courseName varchar(50)
);
create table exam(
  examID int primary key not null,
    courseID int,
    examTitle varchar(50),
    examTime datetime,
    examDuration time
);
create table question(
  questionID int primary key not null,
    examID int,
    questionContent varchar(50)
);
create table answer(
  answerID int primary key not null,
questionID int,
studentID int,
    answerContent varchar(50)
);
alter table answer add constraint fk_answer_student foreign key (studentID) references
student(studentID);
alter table course add constraint fk_course_instructor foreign key (instructorID) references
instructor(instructorID);
alter table participation add constraint fk_participation_student foreign key (studentID)
references student(studentID);
alter table participation add constraint fk_participation_exam foreign key (examID)
references exam(examID);
alter table exam add constraint fk_exam_course foreign key (courseID) references
course(courseID);
alter table question add constraint fk_question_exam foreign key (examID) references
exam(examID);
alter table answer add constraint fk_answer_question foreign key (questionID) references
question(questionID);
CREATE TABLE result (
  answerID INT,
  studentID INT,
  examID INT,
  score INT,
  grade varchar(50),
  PRIMARY KEY (answerID, studentID, examID),
  FOREIGN KEY (answerID) REFERENCES Answer (answerID),
  FOREIGN KEY (studentID) REFERENCES Student (studentID),
  FOREIGN KEY (examID) REFERENCES Exam (examID)
);
CREATE TABLE enroll (
PRIMARY KEY (studentID, courseID),
  studentID INT,
  courseID INT,
enrollDate date,
  FOREIGN KEY (studentID) REFERENCES student(studentID),
  FOREIGN KEY (courseID) REFERENCES course(courseID)
);
```

Data Insertion

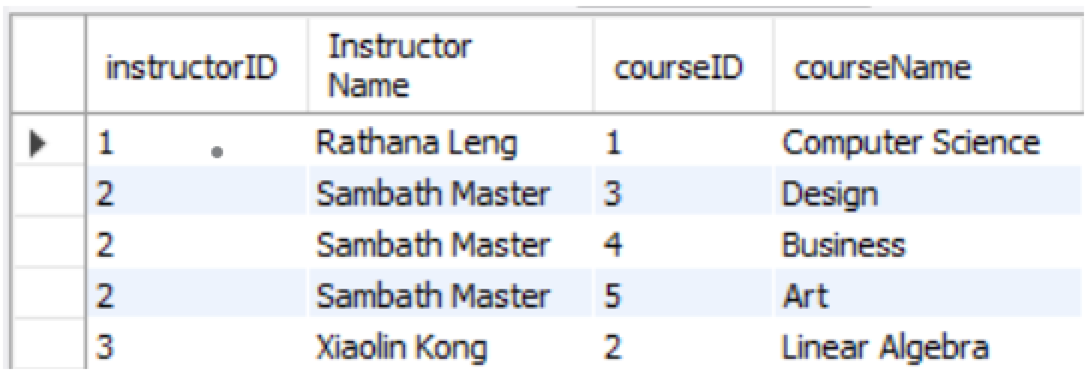
```
insert into
instructor(instructorID,instructorFname,instructorLname,instructor_email,instructor_pw)
values
(1,"Rathana","Leng", "rathanaleng@gmail.com","*********"),
(2,"Sambath","Master", "lgbtq@gmail.com","*********"),
(3,"Xiaolin","Kong", "xiaolinkong@gmail.com","*********");
insert into student (studentID, firstName, lastName, email, password) values
(1,"Jack","Hing","jackhinglol@gmail.com","*********"),
(2,"Rithy","Soy","rithylol@gmail.com","*********"),
(3,"Sovan","Chy","chysovanlol@gmail.com","*********"),
(4,"Hak","MongHout","monghoutlol@gmail.com","*********");
insert into course (courseID,instructorID,courseName) values
(1,1,"Computer Science"),
(2,3,"Linear Algebra"),
(3,2,"Design"),
(4,2,"Business"),
(5,2,"Art");
insert into enroll (studentID, courseID, enrollDate) values
(1,1,'2022-01-01'),
(2,1,'2022-02-10'),
(3,2,'2021-10-01'),
(4,3,'2020-12-12'),
(1,2,'2021-10-20'),
(1,4,'2022-04-01'),
(2,4,'2022-01-01'),
(3,5,'2021-01-01'),
(4,2,'2021-12-12'),
(4,1,'2019-10-12');
insert into exam (examID,courseID,examTitle,examTime,examDuration) values
(1,1,"CS Exam",'2022-10-22 01:00:00','01:00:00'),
(2,2,"LA Exam",'2023-11-20 02:00:00','02:30:00'),
(3,3,"Design Exam",'2023-01-12 3:00:00','00:30:00'),
(4,4,"BUS Exam",'2022-10-22 01:00:00','01:00:00'),
(5,5,"Art Exam",'2023-11-20 02:00:00','02:30:00');
insert into question(questionID,examID,questionContent) values
(1,1,"What is CS?"),
(2,2,"What is 2+1?"),
(3,3,"Describe design"),
(4,4,"How to do business?"),
(5,5,"What is art?");
insert into answer (answerID,questionID,studentID,answerContent) values
(1,1,1,"CS is CS"),
(2,2,1,"2+1 = 3"),
(3,4,1,"I don't know"),
(4,1,2,"CS is idk"),
(5,4,2,"BUS is scamming people"),
(6,2,3,"2+1 = 21"),
(7,5,3,"Art is a way to express yourself"),
(8,1,4,"print('idk')"),
(9,2,4,"2+1 = 12"),
(10,3,4,"Design is creating something");
insert into participation (participationID, studentID, examID, startTime, endTime) values
(1,1,1,'01:00:00', '02:00:00'),
(2,1,2,'02:00:00', '02:30:00'),
(3,1,4,'01:00:00', '01:55:00'),
(4,2,1,'01:00:00', '01:20:00'),
(5,2,4,'01:00:00', '02:25:00'),
(6,3,2,'02:00:00', '03:58:00'),
(7,3,5,'02:00:00', '03:33:00'),
(8,4,1,'01:00:00', '01:15:12'),
(9,4,2,'02:00:00', '03:28:12'),
(10,4,3,'03:00:00', '03:15:08');
insert into result (answerid,studentid,examid,score,grade) values
(1,1,1,99,"A"),
(2,1,2,90,"A"),
(3,1,4,40,"F"),
(4,2,1,60,"C"),
(5,2,4,60,"C"),
(6,3,2,60,"C"),
(7,3,5,21,"F"),
(8,4,1,100,"A"),
(9,4,2,75,"B"),
(10,4,3,80,"B");
```

Chapter 4:

Testing and Evaluation

Select relating queries (2 queries):

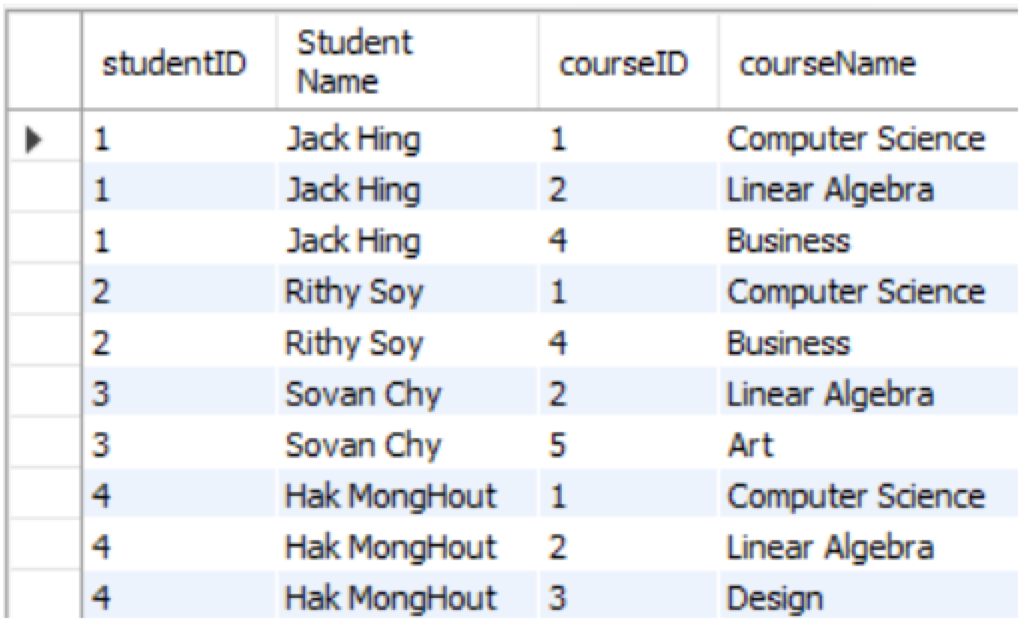
```
select instructor.instructorID, concat(instructorFname, " " , instructorLname) as "Instructor
Name",
courseID, courseName from instructor, course where instructor.instructorID =
course.instructorID;
```



	instructorID	Instructor Name	courseID	courseName
▶	1	Rathana Leng	1	Computer Science
	2	Sambath Master	3	Design
	2	Sambath Master	4	Business
	2	Sambath Master	5	Art
	3	Xiaolin Kong	2	Linear Algebra

In this query we are finding which instructor teaches which course

```
select student.studentID, concat(student.firstName," ",student.lastName) as "Student Name",
course.courseID, course.courseName
from student, course, enroll where
enroll.studentID = student.studentID and
enroll.courseID = course.courseID;
```

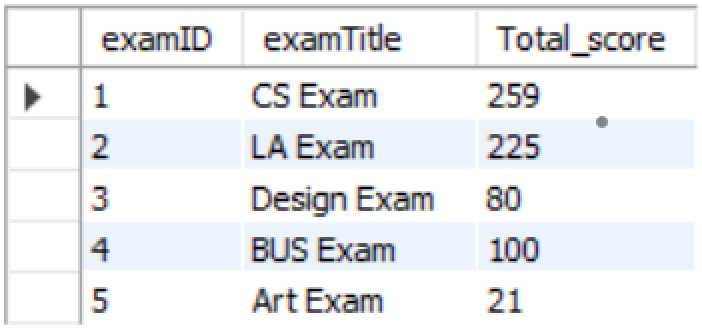


	studentID	Student Name	courseID	courseName
▶	1	Jack Hing	1	Computer Science
	1	Jack Hing	2	Linear Algebra
	1	Jack Hing	4	Business
	2	Rithy Soy	1	Computer Science
	2	Rithy Soy	4	Business
	3	Sovan Chy	2	Linear Algebra
	3	Sovan Chy	5	Art
	4	Hak MongHout	1	Computer Science
	4	Hak MongHout	2	Linear Algebra
	4	Hak MongHout	3	Design

In this query we are displaying the students and their enrolled courses

SUM, COUNT, AVG, MAX, MIN (5 queries):

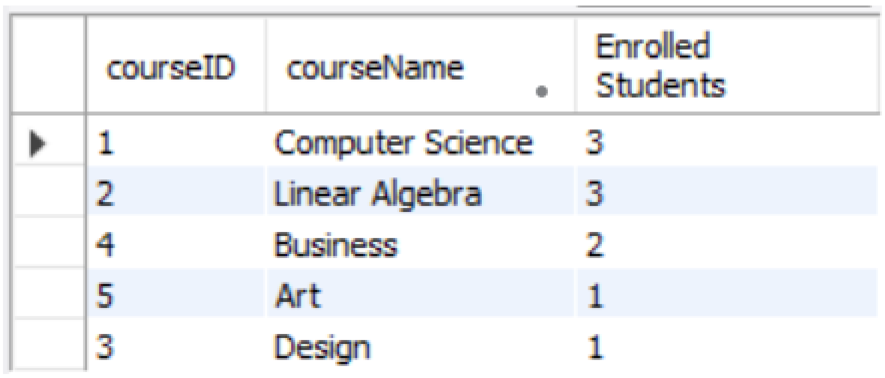
```
select exam.examID, examTitle, sum(score) as Total_score from result
join exam on result.examID = exam.examID group by examID;
```



examID	examTitle	Total_score
1	CS Exam	259
2	LA Exam	225
3	Design Exam	80
4	BUS Exam	100
5	Art Exam	21

In this query we are using SUM() to show the total score for all exams that are collected from students

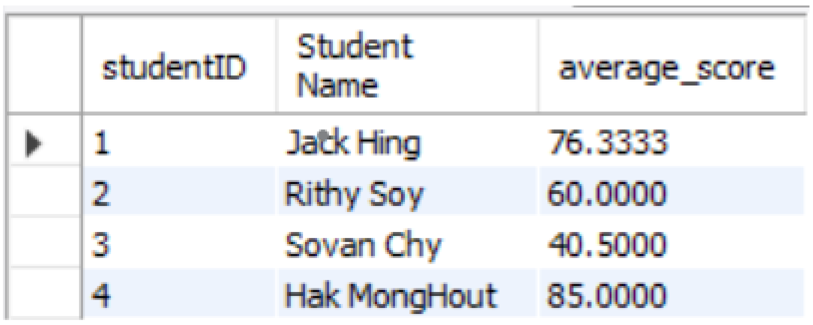
```
select course.courseID, course.courseName,  count(student.studentID) as "Enrolled Students" from
student, course, enroll where
enroll.studentID = student.studentID and
enroll.courseID = course.courseID group by course.courseID;
```



courseID	courseName	Enrolled Students
1	Computer Science	3
2	Linear Algebra	3
4	Business	2
5	Art	1
3	Design	1

In this query we are using COUNT() to count amount of students enrolled in a course

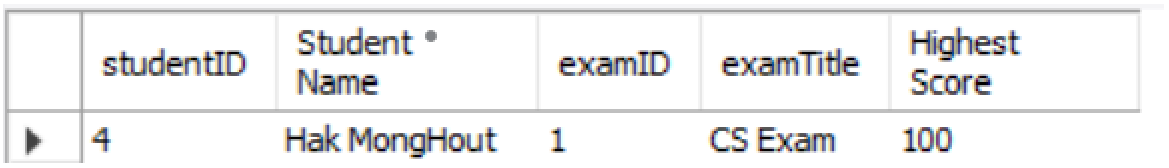
```
select student.studentID,concat(student.firstName, " ", student.lastName) as "Student Name",avg(score) as average_score from result
join student on result.studentID = student.studentID group by studentID;
```



studentID	Student Name	average_score
1	Jack Hing	76.3333
2	Rithy Soy	60.0000
3	Sovan Chy	40.5000
4	Hak MongHout	85.0000

In this query we use AVG() in order to find the average score for the students

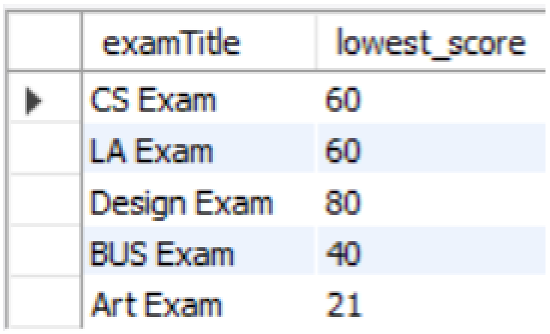
```
select student.studentID, concat(student.firstName," ",student.lastName) as "Student Name",
exam.examID, exam.examTitle, max(score) as "Highest Score"
from student, result, exam where
student.studentID = result.studentID and
exam.examID = result.examID group by studentID, examID
having max(score) = (select max(score) from result where examID = 1);
```



	studentID	Student Name	examID	examTitle	Highest Score
▶	4	Hak MongHout	1	CS Exam	100

In this query we use MAX() in order to determine the highest scored student for an exam

```
select examTitle, min(score) as lowest_score from result, exam
where exam.examID = result.examID group by exam.examID;
```

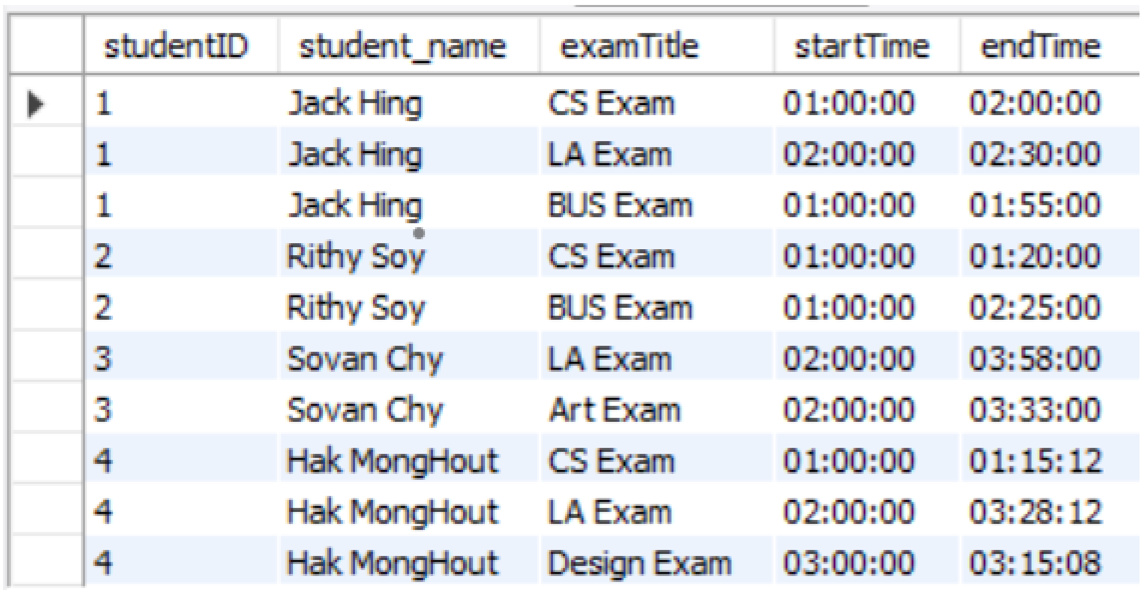


	examTitle	lowest_score
▶	CS Exam	60
	LA Exam	60
	Design Exam	80
	BUS Exam	40
	Art Exam	21

In this query we use MIN() to determine the lowest score for each exam

Join queries (2 queries):

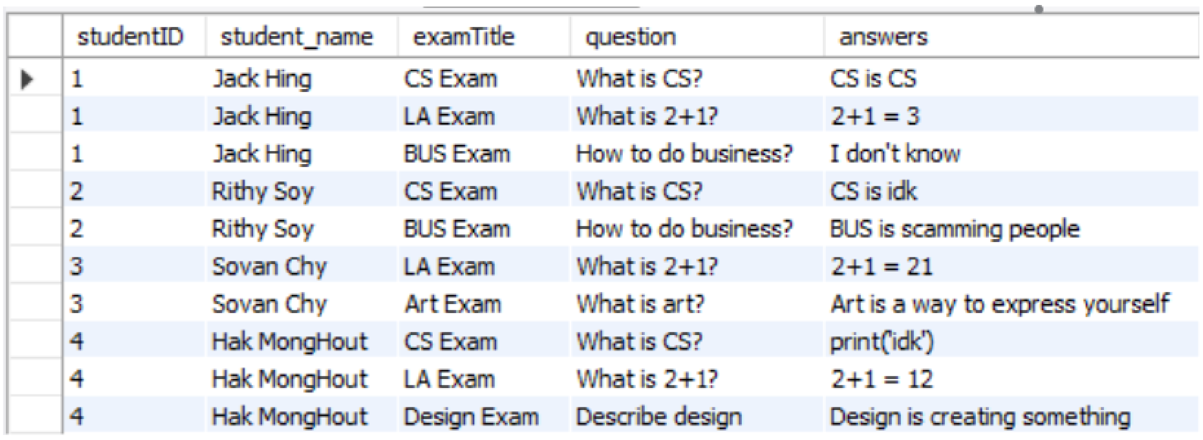
```
select student.studentID, concat(firstName, " ", lastName) as student_name, examTitle,
startTime, endTime from participation
join student on participation.studentID = student.studentID
join exam on participation.examID = exam.examID;
```



studentID	student_name	examTitle	startTime	endTime
1	Jack Hing	CS Exam	01:00:00	02:00:00
1	Jack Hing	LA Exam	02:00:00	02:30:00
1	Jack Hing	BUS Exam	01:00:00	01:55:00
2	Rithy Soy	CS Exam	01:00:00	01:20:00
2	Rithy Soy	BUS Exam	01:00:00	02:25:00
3	Sovan Chy	LA Exam	02:00:00	03:58:00
3	Sovan Chy	Art Exam	02:00:00	03:33:00
4	Hak MongHout	CS Exam	01:00:00	01:15:12
4	Hak MongHout	LA Exam	02:00:00	03:28:12
4	Hak MongHout	Design Exam	03:00:00	03:15:08

In this query we are retrieving all the records for all the student's participation on exams

select student.studentID, concat(firstName, " ", lastName) as student_name, examTitle, questionContent as question, answerContent as answers from answer
join student on answer.studentID = student.studentID
join question on answer.questionID = question.questionID
join exam on question.examID = exam.examID;

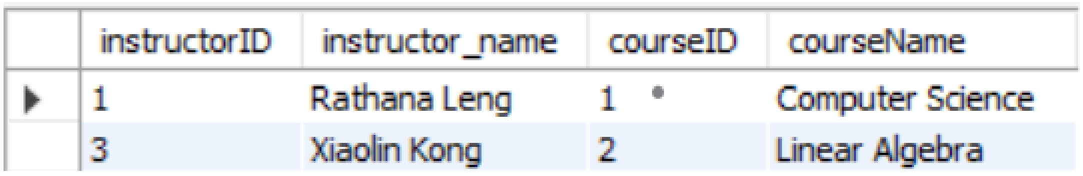


studentID	student_name	examTitle	question	answers
1	Jack Hing	CS Exam	What is CS?	CS is CS
1	Jack Hing	LA Exam	What is 2+1?	2+1 = 3
1	Jack Hing	BUS Exam	How to do business?	I don't know
2	Rithy Soy	CS Exam	What is CS?	CS is idk
2	Rithy Soy	BUS Exam	How to do business?	BUS is scamming people
3	Sovan Chy	LA Exam	What is 2+1?	2+1 = 21
3	Sovan Chy	Art Exam	What is art?	Art is a way to express yourself
4	Hak MongHout	CS Exam	What is CS?	print('idk')
4	Hak MongHout	LA Exam	What is 2+1?	2+1 = 12
4	Hak MongHout	Design Exam	Describe design	Design is creating something

In this query we are retrieving the questions and their corresponding answers by students

Join NOT (1 query):

select instructor.instructorID, concat(instructorFname, " ", instructorLname) as instructor_name, course.courseID, courseName from course
join instructor on course.instructorID = instructor.instructorID
where not instructor.instructorID = 2;

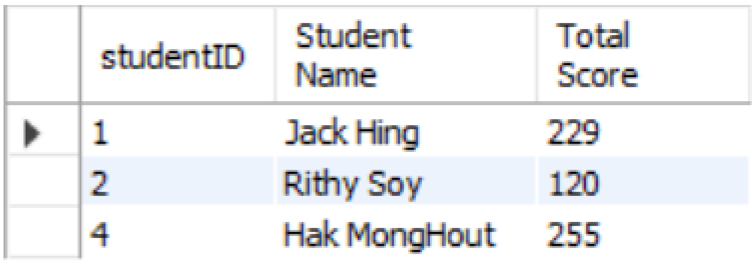


	instructorID	instructor_name	courseID	courseName
▶	1	Rathana Leng	1	Computer Science
	3	Xiaolin Kong	2	Linear Algebra

In this query we use join not to retrieve courses that are not taught by instructorID = 2, which is "Sambath Master"

Group by and Having queries (2 queries):

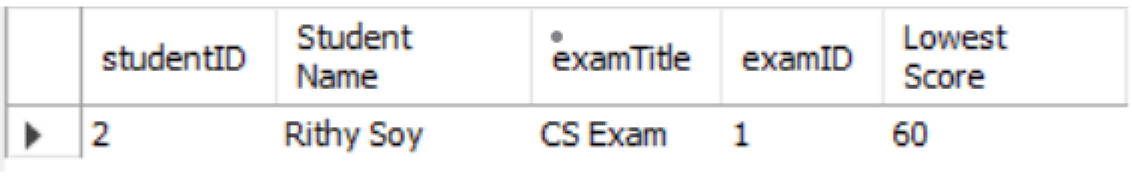
```
select student.studentID, concat(student.firstName," ",student.lastName) as "Student Name",
sum(result.score) as "Total Score"
from student, result where
student.studentID = result.studentID group by studentID
having sum(result.score) >= 100;
```



	studentID	Student Name	Total Score
▶	1	Jack Hing	229
	2	Rithy Soy	120
	4	Hak MongHout	255

In this query we are showing the total score for all students if their score >= 100

```
select student.studentID,concat(firstName," ",lastName) as "Student Name",
exam.examTitle, exam.examID, min(score) as "Lowest Score"
from result
join student ON result.studentID = student.studentID
join exam ON result.examID = exam.examID
where exam.examID = 1
group by student.studentID
having min(score) = (select min(score) from result where examID = 1);
```

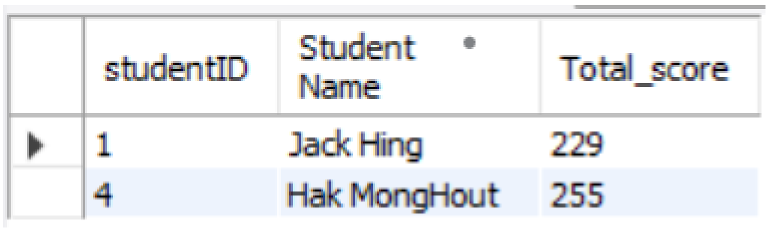


	studentID	Student Name	examTitle	examID	Lowest Score
▶	2	Rithy Soy	CS Exam	1	60

In this query we are showing the lowest score for a specific exam and by which student is it from

Distinct and All queries (2 queries):

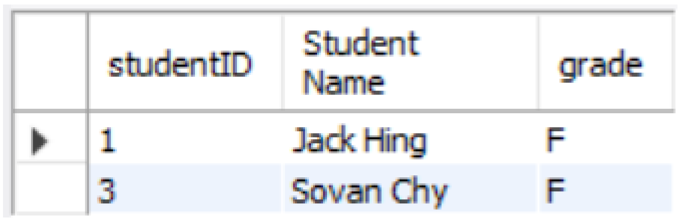
```
select distinct student.studentID, concat(firstName," ",lastName) as "Student Name",
sum(score) as Total_score from result
join student on result.studentID = student.studentID
group by student.studentID
having sum(score) > all (select sum(score) from result where studentID = 2);
```



	studentID	Student Name	Total_score
▶	1	Jack Hing	229
	4	Hak MongHout	255

In this query we display students that gains higher total score than a specific student, in this case studentID = 2 which is "Soy Rithy"

```
select distinct student.studentID, concat(firstName," ",lastName) as "Student Name", grade
from result
join student on result.studentID = student.studentID
having grade > all (select grade from result where studentID = 2);
```



	studentID	Student Name	grade
▶	1	Jack Hing	F
	3	Sovan Chy	F

In this query shows student who is graded lower than studentID = 2 which is "Soy Rithy"
keep in mind that > is higher but since grade is in varchar it actually means that
for example A < B because in ASCII A < B so to find the lower grade we use > which finds the lower character grade

Conclusion

The implementation of this online exam system allows us to evaluate our database design skills and our abilities to implement the design ideas into a real database. By using various methods such as ERD for conceptualizing our design ideas, normalization to clean our data, Data dictionary to outline the data inside the tables, and using SQL queries to create and manage the data, we have shown that we can effectively create a basically working database that can be used as the backend for an app or website.

Future Enhancement:

- Implement a working website fronted using React JS, CSS and HTML
- Use Laravel for database interaction
- Add authentication system