

DATABASE SYSTEM
EXAM SCHEDULING MANAGEMENT SYSTEM

Table of contents

• Introduction	1
• Problem Statement and Objective	3
• Rules	4
• ERD	5
• Normalization	6
• Data Dictionary	9
• SQL Code	10
• Business Ideas	39
• Conclusion.....	40

Introduction

This database system was developed for the Faculty of Computer Science in a small university which did not have a database system setup for scheduling exams. In the coming weeks, various students will be seating for their various exams for this trimester, which are all offered from various different courses. All of which will need to have their own location, date, timings and so on. Thus, to help solve this complex issue, a simple database system was helped to be setup for this particular faculty from this university for this particular trimester, which is currently the first trimester, in order to prepare themselves to handle the scheduling of the upcoming exams.

Invigilator

This university will provide 5 invigilators for the upcoming exams. Each of these invigilators will be in charge of one exam at a time. They are responsible for paper distribution and collection, managing the exam room tables, seating order and will be responsible to look on the students to ensure the exam is done well, without any noise or cheating. Each invigilator will have their unique Invigilator id, which will be their Lecturer ID. The data stored regarding each invigilator will also store their first name and last name.

Course

Every trimester will have their own unique set of courses offered to the students. Since this database is solely focusing on the Faculty of Computer Science, thus, for this semester, the subjects where the students will have to sit for their exams would be Computer Programming, Operating Systems, Mathematical Techniques and Database Systems. The data of each of these courses, which also includes their unique course ID, course name and also their course credits will all be saved and recorded in the database system.

Student

The details of the students whom are enrolled into these courses will also be taken down for organization purposes such as to track their attendance in the exams, marks, to see how many students will be attending the examinations and so on. The data of students that will be recorded will be their unique student ID number, their first and last names along with the students email addresses. The details of the students will only take into account the students whom are enrolled in the Faculty of Computer Science for the first trimester.

Room

The examinations must be held at a few centralized venues in order to ensure all students have a good environment to proceed with their exams, venues that fit the right capacity of students and have the proper facilities such as good lighting, ample of space, air conditioning and so on. There will be two different types of rooms, first are the Lab rooms to handle lab examinations where students will have to use the university monitors to solve their exams and another will be the exam room where students will write down their answers on a given paper. The details of all of this rooms, such as their unique room number and their room type will all be stored in the database system.

Exam

Each course will provide their own sets of examinations and these examinations can either be of different types which either lab, midterm, finals or all of them in different dates. These exams will be held in a way to ensure that there is no clashing of rooms, dates, timings, and invigilators while also providing the students some time for them to be able to revise for the upcoming examinations. Thus, these exams, which will have various types, offered by the courses, will be held in unique dates and times, in a proper room with an invigilator monitoring the students. So, the data that will be collected in the database system would be the unique exam ID, the exam type, exam date, exam start and finish time, the unique course ID of the course that is offering the particular exam, the unique room number of where the exam will be held and the unique invigilator ID of the invigilator that will monitor that particular exam.

Problem Statements & Objectives

Problem Statements	Objectives
There is no proper system to schedule students' exam timetables.	To develop a database system that schedules all student's exam timetables.
Difficulty in resolving conflicts.	To resolve any conflicts between exams scheduled at the same date, time or venue.
Lack of automation causes difficulty in managing exam scheduling.	To improve the overall exam scheduling process and make it more efficient and effective for educational institutions and to provide a user-friendly interface for easy navigation and use.

Rules

1. Each course can offer multiple exams. Each exam will be associated with one course only.
2. Each course can be enrolled by many students. Each student can enrol in many courses.
3. Each exam can take place in one exam room. Each exam room, can only have one exam taking place at a time.
4. Each exam can have many students, and each student can take part in many exams. However, each student can only take one exam at a time.
5. Each exam can be managed by only one invigilator, but each invigilator can manage many exams but only one exam at a time.

Entity Relationship Diagram

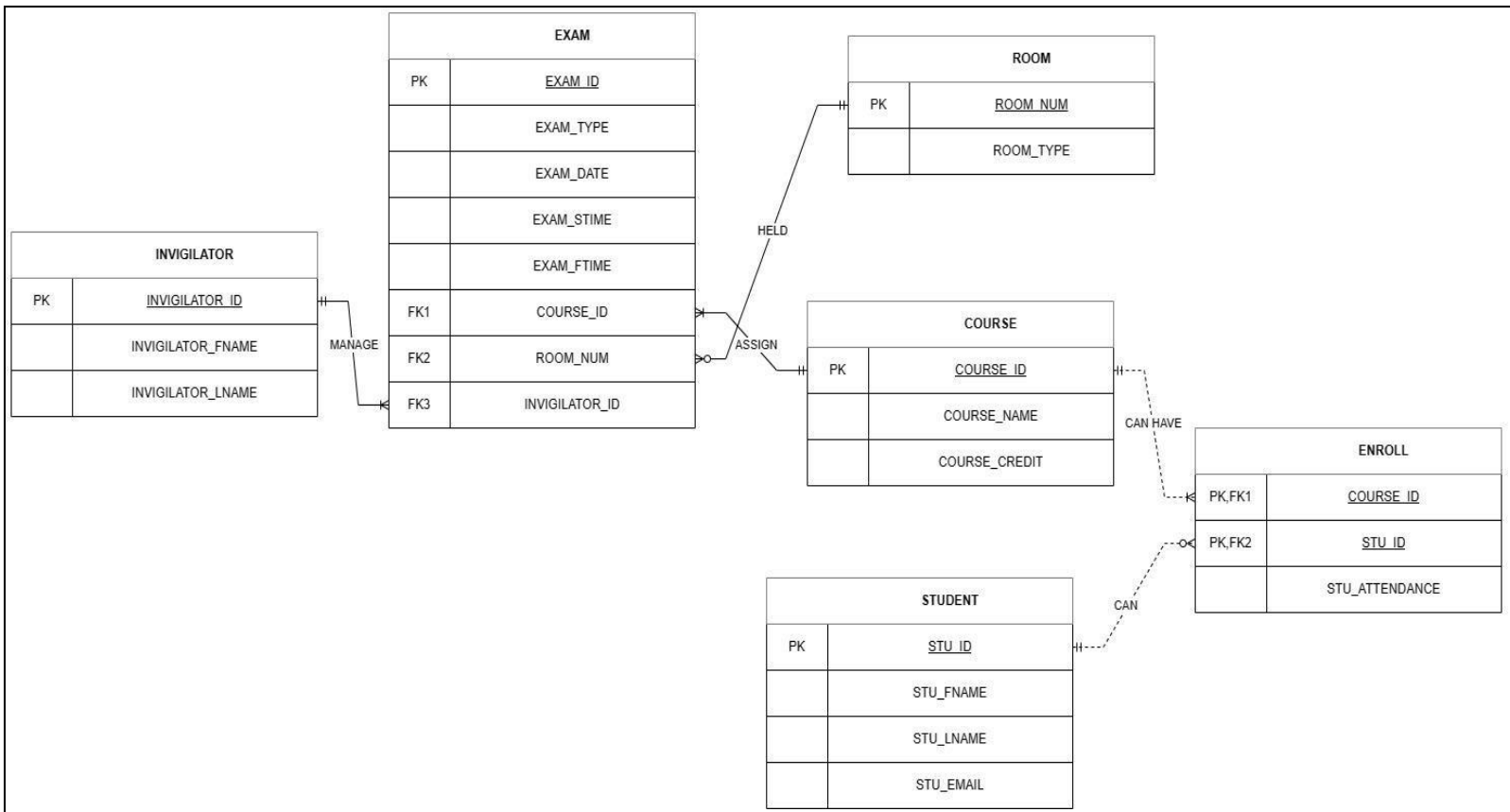


Figure 1. ERD

Normalization

In the first normalization process, there was only one entity found, which is the EXAMS entity.

EXAMS (EXAM_ID, EXAM_TYPE, EXAM_DATE, EXAM_STIME, EXAM_FTIME, STU_NAME, STU_EMAIL, COURSE_ID, COURSE_NAME, COURSE_CREDIT, INVIGILATOR_ID, INVIGILATOR_FNAME, INVIGILATOR_LNAME, ROOM_TYPE)

The following partial dependencies and transitive dependencies were also found during the 1st normalization process from the table EXAMS.

Partial Dependencies:

(EXAM_ID \rightarrow EXAM_TYPE, EXAM_DATE, EXAM_STIME, EXAM_FTIME, ROOM_NAME, ROOM_TYPE)

(INVIGILATOR_ID \rightarrow INVIGILATOR_FNAME, INVIGILATOR_LNAME)

(COURSE_ID \rightarrow COURSE_NAME, COURSE_CREDIT, STU_NAME, STU_EMAIL, STU_ATTENDANCE)

Transitive Dependencies:

(ROOM_NAME \rightarrow ROOM_TYPE)

(STU_NAME \rightarrow STU_EMAIL)

In the 2nd normalization form, once all of the partial dependencies are eliminated, then, there would be 2 additional entities and eventually tables which are all listed below.

EXAM (EXAM_ID, EXAM_TYPE, EXAM_DATE, EXAM_STIME, EXAM_FTIME, ROOM_NAME, ROOM_TYPE)

INVIGILATOR (INVIGILATOR_ID, INVIGILATOR_FNAME, INVIGILATOR_LNAME)

COURSE (COURSE_ID, COURSE_NAME, COURSE_CREDIT, STU_NAME, STU_EMAIL)

In the 3rd Normalization process, once all of the transitive dependencies are eliminated, then they would provide additional tables and entities which are listed below.

STUDENT (STU_NAME, STU_EMAIL)

ROOM (ROOM_NAME, ROOM_TYPE)

However, from the above process, it was found that student name (STU_NAME) and ROOM_NAME from the STUDENT and ROOM tables respectively are not fit enough to be a primary key for those tables as they both may have the exact same name and cannot uniquely identify each other. Thus, brand new attributes such as the student ID (STU_ID) and room number (ROOM_NUM) were added to ensure that both those tables have a primary key of their own. They are listed below.

STUDENT (STU_ID, STU_FNAME, STU_LNAME, STU_EMAIL)

ROOM (ROOM_NUM, ROOM_TYPE)

Meanwhile, some of the previous tables as well from the 2nd normalization form will have their transitive dependencies eliminated in the 3rd normalization process as listed below.

EXAM (EXAM_ID, EXAM_TYPE, EXAM_DATE, EXAM_STIME, EXAM_FTIME)

COURSE (COURSE_ID, COURSE_NAME, COURSE_CREDIT)

**Note:* From the above ERD, it was discovered that the entities STUDENT and COURSE have a many-to-many relationship and thus, to overcome this problem, a composite entity named ENROLL was built as such. A STU_ATTENDANCE attribute was added to also monitor the student's attendance and check on their eligibility to seat for the exams.

ENROLL (COURSE_ID (FK), STU_ID (FK), STU_ATTENDANCE)

Data Dictionary

Table	Attribute	Attribute Description	Type	Format	Range	PK/FK	FK reference
INVIGILATOR	INVIGILATOR_ID	ID of invigilator	INT	9999	0000 - 9999	PK	
	INVIGILATOR_FNAME	First name of invigilator	VARCHAR(30)	XXXX			
	INVIGILATOR_LNAME	Last name of invigilator	VARCHAR(30)	XXXX			
EXAM	EXAM_ID	ID of exam	VARCHAR (10)	XXX999			
	EXAM_TYPE	Lab/Midterm/Final exam	CHAR (10)	XXXX			
	EXAM_DATE	Date of exam	DATE	YYYY-MM-DD			
	EXAM_STIME	Starting time of exam	TIME	HH:MM:SS			
	EXAM_FTIME	Finish time of exam	TIME	HH:MM:SS			
	COURSE_ID	ID of course	VARCHAR (20)	XXX999		FK	From COURSE
	ROOM_NUM	Exam room number	INT	9999	0000 - 9999	FK	From ROOM
	INVIGILATOR_ID	ID of invigilator	INT	9999	0000 - 9999	FK	From INVIGILATOR
ROOM	ROOM_NUM	Exam room number	INT	9999	0000 - 9999	PK	
	ROOM_TYPE	Lab/Exam Room	VARCHAR (15)	XXXX			
COURSE	COURSE_ID	ID of course	VARCHAR (20)	XXX999		PK	
	COURSE_NAME	Name of course	VARCHAR (30)	XXXX			
	COURSE_CREDIT	Number of course credits	INT	9999	00 - 10		
STUDENT	STU_ID	ID of student	INT	9999	0000 - 9999	PK	
	STU_FNAME	Student's first name	VARCHAR (20)	XXXX			
	STU_LNAME	Student's last name	VARCHAR (20)	XXXX			
	STU_EMAIL	Student's email address	VARCHAR (20)	999@XXX.XX.XX			
ENROLL	COURSE_ID	ID of course	VARCHAR (20)	XXX999		FK	From COURSE
	STU_ID	ID of student	INT	9999	0000 - 9999	FK	From STUDENT
	STU_ATTENDANCE	Attendance % of students	DECIMAL (4,2)	99.99	00.00 - 100.00		

Table 1. Data Dictionary

SQL and Screenshots

- a) The next command is to create a database named “PROJECT” and to use it.

CREATE DATABASE PROJECT;

USE PROJECT;

```
XAMPP for Windows - mysql -u root

Setting environment for using XAMPP for Windows.
Tahmida@LAPTOP-27RL05KV c:\xampp
# cd mysql

Tahmida@LAPTOP-27RL05KV c:\xampp\mysql
# cd bin

Tahmida@LAPTOP-27RL05KV c:\xampp\mysql\bin
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1823
Server version: 10.4.27-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE PROJECT;
Query OK, 1 row affected (0.001 sec)

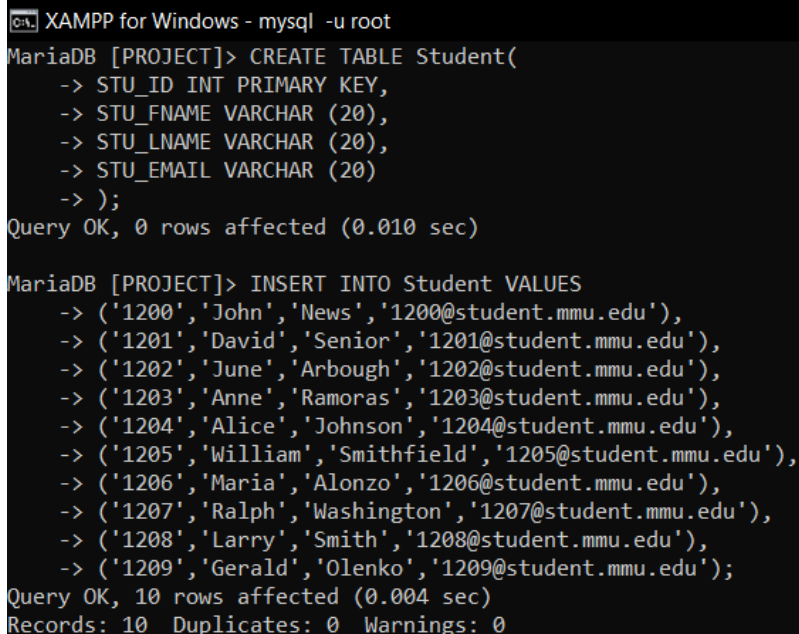
MariaDB [(none)]> USE PROJECT;
Database changed
MariaDB [PROJECT]> 
```

b) The next commands are to create and insert values into the respective tables.

(*STUDENT* table)

```
CREATE TABLE Student(  
STU_ID INT PRIMARY KEY,  
STU_FNAME VARCHAR (20),  
STU_LNAME VARCHAR (20),  
STU_EMAIL VARCHAR (20)  
);
```

```
INSERT INTO Student VALUES  
( '1200', 'John', 'News', '1200@student.mmu.edu'),  
( '1201', 'David', 'Senior', '1201@student.mmu.edu'),  
( '1202', 'June', 'Arbough', '1202@student.mmu.edu'),  
( '1203', 'Anne', 'Ramoras', '1203@student.mmu.edu'),  
( '1204', 'Alice', 'Johnson', '1204@student.mmu.edu'),  
( '1205', 'William', 'Smithfield', '1205@student.mmu.edu'),  
( '1206', 'Maria', 'Alonzo', '1206@student.mmu.edu'),  
( '1207', 'Ralph', 'Washington', '1207@student.mmu.edu'),  
( '1208', 'Larry', 'Smith', '1208@student.mmu.edu'),  
( '1209', 'Gerald', 'Olenko', '1209@student.mmu.edu');
```



```
C:\XAMPP for Windows - mysql -u root  
MariaDB [PROJECT]> CREATE TABLE Student(  
-> STU_ID INT PRIMARY KEY,  
-> STU_FNAME VARCHAR (20),  
-> STU_LNAME VARCHAR (20),  
-> STU_EMAIL VARCHAR (20)  
-> );  
Query OK, 0 rows affected (0.010 sec)  
  
MariaDB [PROJECT]> INSERT INTO Student VALUES  
-> ( '1200', 'John', 'News', '1200@student.mmu.edu'),  
-> ( '1201', 'David', 'Senior', '1201@student.mmu.edu'),  
-> ( '1202', 'June', 'Arbough', '1202@student.mmu.edu'),  
-> ( '1203', 'Anne', 'Ramoras', '1203@student.mmu.edu'),  
-> ( '1204', 'Alice', 'Johnson', '1204@student.mmu.edu'),  
-> ( '1205', 'William', 'Smithfield', '1205@student.mmu.edu'),  
-> ( '1206', 'Maria', 'Alonzo', '1206@student.mmu.edu'),  
-> ( '1207', 'Ralph', 'Washington', '1207@student.mmu.edu'),  
-> ( '1208', 'Larry', 'Smith', '1208@student.mmu.edu'),  
-> ( '1209', 'Gerald', 'Olenko', '1209@student.mmu.edu');  
Query OK, 10 rows affected (0.004 sec)  
Records: 10 Duplicates: 0 Warnings: 0
```

(Course table)

```
CREATE TABLE Course(  
  COURSE_ID VARCHAR(20) PRIMARY KEY,  
  COURSE_NAME VARCHAR(30),  
  COURSE_CREDIT INT  
);
```

```
INSERT INTO Course VALUES  
( 'TCP1121','Computer Programming',4),  
( 'TDB1131','Database Systems',3 ),  
( 'TMA1111','Mathematical Techniques',4 ),  
( 'TOS1141','Operating Systems',3);
```

```
MariaDB [PROJECT]> CREATE TABLE Course(  
  -> COURSE_ID VARCHAR(20) PRIMARY KEY,  
  -> COURSE_NAME VARCHAR(30),  
  -> COURSE_CREDIT INT  
  -> );  
Query OK, 0 rows affected (0.010 sec)  
  
MariaDB [PROJECT]> INSERT INTO Course VALUES  
  -> ( 'TCP1121','Computer Programming',4),  
  -> ( 'TDB1131','Database Systems',3 ),  
  -> ( 'TMA1111','Mathematical Techniques',4 ),  
  -> ( 'TOS1141','Operating Systems',3);  
Query OK, 4 rows affected (0.004 sec)  
Records: 4  Duplicates: 0  Warnings: 0
```

(INVIGILATOR table)

```
CREATE TABLE Invigilator (  
INVIGILATOR_ID INT PRIMARY KEY,  
INVIGILATOR_FNAME VARCHAR (30),  
INVIGILATOR_LNAME VARCHAR (30)  
);
```

```
INSERT INTO Invigilator VALUES  
( '1001','DAVID','WALLACE'),  
( '1002','JAN', 'LEVISON'),  
( '1003','MICHEAL','SCOTT'),  
( '1004','ANGELA','MARTIN'),  
( '1005','STANLEY','HUDSON');
```

```
MariaDB [PROJECT]> CREATE TABLE Invigilator(  
-> INVIGILATOR_ID INT PRIMARY KEY,  
-> INVIGILATOR_FNAME VARCHAR(30),  
-> INVIGILATOR_LNAME VARCHAR(30)  
-> );  
Query OK, 0 rows affected (0.011 sec)  
  
MariaDB [PROJECT]> INSERT INTO Invigilator VALUES  
-> ( '1001','DAVID','WALLACE'),  
-> ( '1002','JAN', 'LEVISON'),  
-> ( '1003','MICHEAL','SCOTT'),  
-> ( '1004','ANGELA','MARTIN'),  
-> ( '1005','STANLEY','HUDSON');  
Query OK, 5 rows affected (0.004 sec)  
Records: 5 Duplicates: 0 Warnings: 0
```

(*ROOM* table)

```
CREATE TABLE Room (  
ROOM_NUM INT PRIMARY KEY,  
ROOM_TYPE VARCHAR (15));
```

```
INSERT INTO Room VALUES  
(101,'LAB'),  
(102,'EXAM-ROOM'),  
(103,'LAB'),  
(104,'EXAM-ROOM');
```

```
MariaDB [PROJECT]> CREATE TABLE Room(  
-> ROOM_NUM INT PRIMARY KEY,  
-> ROOM_TYPE VARCHAR(15)  
-> );  
Query OK, 0 rows affected (0.009 sec)  
  
MariaDB [PROJECT]> INSERT INTO Room VALUES  
-> (101,'LAB'),  
-> (102,'EXAM-ROOM'),  
-> (103,'LAB'),  
-> (104,'EXAM-ROOM');  
Query OK, 4 rows affected (0.003 sec)  
Records: 4 Duplicates: 0 Warnings: 0
```


(*ENROLL* table)

```
CREATE TABLE Enroll(  
  
  COURSE_ID VARCHAR (20),  
  STU_ID INT,  
  STU_ATTENDANCE DECIMAL (4,2),  
  PRIMARY KEY (COURSE_ID, STU_ID),  
  FOREIGN KEY (COURSE_ID) REFERENCES Course (COURSE_ID) ON  
  DELETE CASCADE,  
  FOREIGN KEY (STU_ID) REFERENCES Student (STU_ID) ON DELETE  
  CASCADE  
);
```

```
INSERT INTO Enroll VALUES  
('TMA1111',1200,'60'),  
('TMA1111',1202,'78.89'),  
('TMA1111',1205,'45'),  
('TMA1111',1206,'80'),  
('TMA1111',1203,'85'),  
('TMA1111',1208,'88'),  
('TMA1111',1201,'83.11'),  
('TMA1111',1209,'80'),  
('TCP1121',1200,'80'),  
('TCP1121',1205,'88'),  
('TCP1121',1203,'84'),  
('TCP1121',1201,'89.90'),  
('TCP1121',1208,'81'),  
('TDB1131',1200,'82'),  
('TDB1131',1201,'67'),
```

('TDB1131',1202,'88'),
('TDB1131',1203,'97'),
('TDB1131',1204,'100'),
('TDB1131',1205,'94'),
('TDB1131',1206,'83.33'),
('TDB1131',1207,'76'),
('TDB1131',1208,'66.67'),
('TDB1131',1209,'90'),
('TOS1141',1208,'80'),
('TOS1141',1204,'84'),
('TOS1141',1202,'90.02'),
('TOS1141',1201,'78.65'),
('TOS1141',1206,'56'),
('TOS1141',1209,'94'),
('TOS1141',1205,'100');

```

MariaDB [PROJECT]> CREATE TABLE Enroll(
  -> COURSE_ID VARCHAR(20),
  -> STU_ID INT,
  -> STU_ATTENDANCE DECIMAL (4,2),
  -> PRIMARY KEY (COURSE_ID,STU_ID),
  -> FOREIGN KEY (COURSE_ID) REFERENCES Course(COURSE_ID) ON DELETE CASCADE,
  -> FOREIGN KEY (STU_ID) REFERENCES Student(STU_ID) ON DELETE CASCADE
  -> );
Query OK, 0 rows affected (0.021 sec)

MariaDB [PROJECT]> INSERT INTO Enroll VALUES
  -> ('TMA1111',1200,'60'),
  -> ('TMA1111',1202,'78.89'),
  -> ('TMA1111',1205,'45'),
  -> ('TMA1111',1206,'80'),
  -> ('TMA1111',1203,'85'),
  -> ('TMA1111',1208,'88'),
  -> ('TMA1111',1201,'83.11'),
  -> ('TMA1111',1209,'80'),
  -> ('TCP1121',1200,'80'),
  -> ('TCP1121',1205,'88'),
  -> ('TCP1121',1203,'84'),
  -> ('TCP1121',1201,'89.90'),
  -> ('TCP1121',1208,'81'),
  -> ('TDB1131',1200,'82'),
  -> ('TDB1131',1201,'67'),
  -> ('TDB1131',1202,'88'),
  -> ('TDB1131',1203,'97'),
  -> ('TDB1131',1204,'100'),
  -> ('TDB1131',1205,'94'),
  -> ('TDB1131',1206,'83.33'),
  -> ('TDB1131',1207,'76'),
  -> ('TDB1131',1208,'66.67'),
  -> ('TDB1131',1209,'90'),
  -> ('TOS1141',1208,'80'),
  -> ('TOS1141',1204,'84'),
  -> ('TOS1141',1202,'90.02'),
  -> ('TOS1141',1201,'78.65'),
  -> ('TOS1141',1206,'56'),
  -> ('TOS1141',1209,'94'),
  -> ('TOS1141',1205,'100');
Query OK, 30 rows affected, 2 warnings (0.009 sec)
Records: 30 Duplicates: 0 Warnings: 2

```

(*EXAM* table)

```
CREATE TABLE Exam (  
EXAM_ID VARCHAR (10) PRIMARY KEY,  
EXAM_TYPE CHAR (10),  
EXAM_DATE DATE,  
EXAM_STIME TIME,  
EXAM_FTIME TIME,  
COURSE_ID VARCHAR (20),  
ROOM_NUM INT,  
INVIGILATOR_ID INT,  
FOREIGN KEY (COURSE_ID) REFERENCES Course (COURSE_ID) ON  
DELETE CASCADE,  
FOREIGN KEY (ROOM_NUM) REFERENCES Room (ROOM_NUM) ON  
DELETE SET NULL,  
FOREIGN KEY (INVIGILATOR_ID) REFERENCES Invigilator  
(INVIGILATOR_ID) ON DELETE SET NULL  
);
```

INSERT INTO Exam VALUES

```
('MAMID111','MIDTERM','2023-01-05','20:00:00','21:30:00','TMA1111',104,1001),
('MAFI112','FINAL','2023-01-30','10:00:00','12:00:00','TMA1111',102,1003),
('CPMID121','LAB','2023-01-12','20:00:00','21:30:00','TCP1121',101,1005),
('CPFI121','FINAL','2023-02-06','10:00:00','12:00:00','TCP1121',102,1005),
('DBMID131','MIDTERM','2023-01-03','20:00:00','21:30:00','TDB1131',102,1001),
('DBFI131','FINAL','2023-02-15','10:00:00','12:00:00','TDB1131',104,1004),
('OSMID141','MIDTERM','2023-01-11','20:00:00','21:30:00','TOS1141',104,1003),
('OSFI141','FINAL','2023-01-30','10:00:00','12:00:00','TOS1141',102,1002);
```

XAMPP for Windows - mysql -u root

MariaDB [project]> CREATE TABLE Exam(

```
-> EXAM_ID VARCHAR(10) PRIMARY KEY,
-> EXAM_TYPE CHAR(10),
-> EXAM_DATE DATE,
-> EXAM_STIME TIME,
-> EXAM_FTIME TIME,
-> COURSE_ID VARCHAR(20),
-> ROOM_NUM INT,
-> INVIGILATOR_ID INT,
-> FOREIGN KEY (COURSE_ID) REFERENCES Course(COURSE_ID) ON DELETE CASCADE,
-> FOREIGN KEY (ROOM_NUM) REFERENCES Room(ROOM_NUM) ON DELETE SET NULL,
-> FOREIGN KEY (INVIGILATOR_ID) REFERENCES Invigilator(INVIGILATOR_ID) ON DELETE SET NULL
-> );
```

Query OK, 0 rows affected (0.024 sec)

MariaDB [project]> INSERT INTO Exam VALUES

```
-> ('MAMID111','MIDTERM','2023-01-05','20:00:00','21:30:00','TMA1111',104,1001),
-> ('MAFI112','FINAL','2023-01-30','10:00:00','12:00:00','TMA1111',102,1003),
-> ('CPMID121','LAB','2023-01-12','20:00:00','21:30:00','TCP1121',101,1005),
-> ('CPFI121','FINAL','2023-02-06','10:00:00','12:00:00','TCP1121',102,1005),
-> ('DBMID131','MIDTERM','2023-01-03','20:00:00','21:30:00','TDB1131',102,1001),
-> ('DBFI131','FINAL','2023-02-15','10:00:00','12:00:00','TDB1131',104,1004),
-> ('OSMID141','MIDTERM','2023-01-11','20:00:00','21:30:00','TOS1141',104,1003),
-> ('OSFI141','FINAL','2023-01-30','10:00:00','12:00:00','TOS1141',102,1002);
```

Query OK, 8 rows affected (0.004 sec)

- The next commands to show every table that has been created in the database using the SELET*FROM command.

SELECT * FROM Student;

```
XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> SELECT * FROM Student;
```

STU_ID	STU_FNAME	STU_LNAME	STU_EMAIL
1200	John	News	1200@student.mmu.edu
1201	David	Senior	1201@student.mmu.edu
1202	June	Arbough	1202@student.mmu.edu
1203	Anne	Ramoras	1203@student.mmu.edu
1204	Alice	Johnson	1204@student.mmu.edu
1205	William	Smithfield	1205@student.mmu.edu
1206	Maria	Alonzo	1206@student.mmu.edu
1207	Ralph	Washington	1207@student.mmu.edu
1208	Larry	Smith	1208@student.mmu.edu
1209	Gerald	Olenko	1209@student.mmu.edu

10 rows in set (0.001 sec)

SELECT * FROM Course;

```
MariaDB [PROJECT]> SELECT * FROM Course;
```

COURSE_ID	COURSE_NAME	COURSE_CREDIT
TCP1121	Computer Programming	4
TDB1131	Database Systems	3
TMA1111	Mathematical Techniques	4
TOS1141	Operating Systems	3

4 rows in set (0.001 sec)

SELECT * FROM Invigilator;

```
MariaDB [PROJECT]> SELECT * FROM Invigilator;
```

INVIGILATOR_ID	INVIGILATOR_FNAME	INVIGILATOR_LNAME
1001	DAVID	WALLACE
1002	JAN	LEVISON
1003	MICHEAL	SCOTT
1004	ANGELA	MARTIN
1005	STANLEY	HUDSON

5 rows in set (0.001 sec)

SELECT * FROM Room;

```
MariaDB [PROJECT]> SELECT * FROM Room;
```

ROOM_NUM	ROOM_TYPE
101	LAB
102	EXAM-ROOM
103	LAB
104	EXAM-ROOM

4 rows in set (0.001 sec)

SELECT * FROM Enroll;

```
MariaDB [PROJECT]> SELECT * FROM Enroll;
```

COURSE_ID	STU_ID	STU_ATTENDANCE
TCP1121	1200	80.00
TCP1121	1201	89.90
TCP1121	1203	84.00
TCP1121	1205	88.00
TCP1121	1208	81.00
TDB1131	1200	82.00
TDB1131	1201	67.00
TDB1131	1202	88.00
TDB1131	1203	97.00
TDB1131	1204	99.99
TDB1131	1205	94.00
TDB1131	1206	83.33
TDB1131	1207	76.00
TDB1131	1208	66.67
TDB1131	1209	90.00
TMA1111	1200	60.00
TMA1111	1201	83.11
TMA1111	1202	78.89
TMA1111	1203	85.00
TMA1111	1205	45.00
TMA1111	1206	80.00
TMA1111	1208	88.00
TMA1111	1209	80.00
TOS1141	1201	78.65
TOS1141	1202	90.02
TOS1141	1204	84.00
TOS1141	1205	99.99
TOS1141	1206	56.00
TOS1141	1208	80.00
TOS1141	1209	94.00

30 rows in set (0.001 sec)

SELECT * FROM Exam;

MariaDB [project]> SELECT * FROM Exam;

EXAM_ID	EXAM_TYPE	EXAM_DATE	EXAM_STIME	EXAM_FTIME	COURSE_ID	ROOM_NUM	INVIGILATOR_ID
CPFI121	FINAL	2023-02-06	10:00:00	12:00:00	TCP1121	102	1005
CPMID121	LAB	2023-01-12	20:00:00	21:30:00	TCP1121	101	1005
DBFI131	FINAL	2023-02-15	10:00:00	12:00:00	TDB1131	104	1004
DBMID131	MIDTERM	2023-01-03	20:00:00	21:30:00	TDB1131	102	1001
MAFI112	FINAL	2023-01-30	10:00:00	12:00:00	TMA1111	102	1003
MAMID111	MIDTERM	2023-01-05	20:00:00	21:30:00	TMA1111	104	1001
OSFI141	FINAL	2023-01-30	10:00:00	12:00:00	TOS1141	102	1002
OSMID141	MIDTERM	2023-01-11	20:00:00	21:30:00	TOS1141	104	1003

8 rows in set (0.000 sec)

- The next command is to alter the table ENROLL to include a new column named STATUS that shows the students eligibility.

ALTER TABLE Enroll

ADD COLUMN STATUS VARCHAR (20) DEFAULT 'NULL';

SELECT * FROM Enroll;

BEFORE:

```
C:\XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> ALTER TABLE Enroll
    -> ADD COLUMN STATUS VARCHAR(20) DEFAULT 'NULL';
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [PROJECT]> SELECT*FROM Enroll;
```

COURSE_ID	STU_ID	STU_ATTENDANCE	STATUS
TCP1121	1200	80.00	NULL
TCP1121	1201	89.90	NULL
TCP1121	1203	84.00	NULL
TCP1121	1205	88.00	NULL
TCP1121	1208	81.00	NULL
TDB1131	1200	82.00	NULL
TDB1131	1201	67.00	NULL
TDB1131	1202	88.00	NULL
TDB1131	1203	97.00	NULL
TDB1131	1204	99.99	NULL
TDB1131	1205	94.00	NULL
TDB1131	1206	83.33	NULL
TDB1131	1207	76.00	NULL
TDB1131	1208	66.67	NULL
TDB1131	1209	90.00	NULL
TMA1111	1200	60.00	NULL
TMA1111	1201	83.11	NULL
TMA1111	1202	78.89	NULL
TMA1111	1203	85.00	NULL
TMA1111	1205	45.00	NULL
TMA1111	1206	80.00	NULL
TMA1111	1208	88.00	NULL
TMA1111	1209	80.00	NULL
TOS1141	1201	78.65	NULL
TOS1141	1202	90.02	NULL
TOS1141	1204	84.00	NULL
TOS1141	1205	99.99	NULL
TOS1141	1206	56.00	NULL
TOS1141	1208	80.00	NULL
TOS1141	1209	94.00	NULL

```
30 rows in set (0.001 sec)
```

```

UPDATE Enroll
SET STATUS = 'ELIGIBLE'
WHERE STU_ATTENDANCE >= '80.00';
SELECT * FROM Enroll;

```

AFTER:

```

MariaDB [PROJECT]> UPDATE Enroll
  -> SET STATUS = 'ELIGIBLE'
  -> WHERE STU_ATTENDANCE >= '80.00';
Query OK, 22 rows affected (0.004 sec)
Rows matched: 22  Changed: 22  Warnings: 0

MariaDB [PROJECT]> SELECT*FROM Enroll;
+-----+-----+-----+-----+
| COURSE_ID | STU_ID | STU_ATTENDANCE | STATUS |
+-----+-----+-----+-----+
| TCP1121   | 1200   | 80.00          | ELIGIBLE |
| TCP1121   | 1201   | 89.90          | ELIGIBLE |
| TCP1121   | 1203   | 84.00          | ELIGIBLE |
| TCP1121   | 1205   | 88.00          | ELIGIBLE |
| TCP1121   | 1208   | 81.00          | ELIGIBLE |
| TDB1131   | 1200   | 82.00          | ELIGIBLE |
| TDB1131   | 1201   | 67.00          | NULL |
| TDB1131   | 1202   | 88.00          | ELIGIBLE |
| TDB1131   | 1203   | 97.00          | ELIGIBLE |
| TDB1131   | 1204   | 99.99          | ELIGIBLE |
| TDB1131   | 1205   | 94.00          | ELIGIBLE |
| TDB1131   | 1206   | 83.33          | ELIGIBLE |
| TDB1131   | 1207   | 76.00          | NULL |
| TDB1131   | 1208   | 66.67          | NULL |
| TDB1131   | 1209   | 90.00          | ELIGIBLE |
| TMA1111   | 1200   | 60.00          | NULL |
| TMA1111   | 1201   | 83.11          | ELIGIBLE |
| TMA1111   | 1202   | 78.89          | NULL |
| TMA1111   | 1203   | 85.00          | ELIGIBLE |
| TMA1111   | 1205   | 45.00          | NULL |
| TMA1111   | 1206   | 80.00          | ELIGIBLE |
| TMA1111   | 1208   | 88.00          | ELIGIBLE |
| TMA1111   | 1209   | 80.00          | ELIGIBLE |
| TOS1141   | 1201   | 78.65          | NULL |
| TOS1141   | 1202   | 90.02          | ELIGIBLE |
| TOS1141   | 1204   | 84.00          | ELIGIBLE |
| TOS1141   | 1205   | 99.99          | ELIGIBLE |
| TOS1141   | 1206   | 56.00          | NULL |
| TOS1141   | 1208   | 80.00          | ELIGIBLE |
| TOS1141   | 1209   | 94.00          | ELIGIBLE |
+-----+-----+-----+-----+
30 rows in set (0.001 sec)

```

- The next command is to delete the room 103 from the table ROOM.

DELETE FROM Room

WHERE ROOM_NUM = 103;

```
C:\xampp> XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> SELECT * FROM Room;
+-----+-----+
| ROOM_NUM | ROOM_TYPE |
+-----+-----+
| 101 | LAB |
| 102 | EXAM-ROOM |
| 103 | LAB |
| 104 | EXAM-ROOM |
+-----+-----+
4 rows in set (0.000 sec)

MariaDB [PROJECT]> DELETE FROM Room
-> WHERE ROOM_NUM = 103;
Query OK, 1 row affected (0.004 sec)

MariaDB [PROJECT]> SELECT * FROM Room;
+-----+-----+
| ROOM_NUM | ROOM_TYPE |
+-----+-----+
| 101 | LAB |
| 102 | EXAM-ROOM |
| 104 | EXAM-ROOM |
+-----+-----+
3 rows in set (0.000 sec)
```

- The next command is to show all of the exams with the respective courses using the JOIN command.

```
SELECT Exam.EXAM_ID , Exam.EXAM_TYPE, Course.COURSE_ID
FROM Exam
JOIN
Course
ON Exam.COURSE_ID = Course.COURSE_ID;
```

```
XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> SELECT Exam.EXAM_ID , Exam.EXAM_TYPE, Course.COURSE_ID
-> FROM Exam
-> JOIN
-> Course
-> ON Exam.COURSE_ID = Course.COURSE_ID;
```

EXAM_ID	EXAM_TYPE	COURSE_ID
CPFI121	FINAL	TCP1121
CPMID121	LAB	TCP1121
DBFI131	FINAL	TDB1131
DBMID131	MIDTERM	TDB1131
MAFI112	FINAL	TMA1111
MAMID111	MIDTERM	TMA1111
OSFI141	FINAL	TDB1131
OSMID141	MIDTERM	TDB1131

```
8 rows in set (0.001 sec)
```

- The next command is to show all of the exams and the respective invigilators that will be assigned to them.

SELECT

**Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STI
ME,Exam.EXAM_FTIME,**

**Invigilator.INVIGILATOR_ID, Invigilator.INVIGILATOR_FNAME,
INVIGILATOR_LNAME**

FROM Exam,Invigilator

WHERE Exam.INVIGILATOR_ID = Invigilator.INVIGILATOR_ID;

```

XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> SELECT Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STIME,Exam.EXAM_FTIME,
-> Invigilator.INVIGILATOR_ID, Invigilator.INVIGILATOR_FNAME, INVIGILATOR_LNAME
-> FROM Exam,Invigilator
-> WHERE Exam.INVIGILATOR_ID = Invigilator.INVIGILATOR_ID;

```

EXAM_ID	EXAM_TYPE	EXAM_DATE	EXAM_STIME	EXAM_FTIME	INVIGILATOR_ID	INVIGILATOR_FNAME	INVIGILATOR_LNAME
CPFI121	FINAL	2023-02-06	10:00:00	12:00:00	1005	STANLEY	HUDSON
CPMID121	LAB	2023-01-12	20:00:00	21:30:00	1005	STANLEY	HUDSON
DBFI131	FINAL	2023-02-15	10:00:00	12:00:00	1004	ANGELA	MARTIN
DBMID131	MIDTERM	2023-01-03	20:00:00	21:30:00	1001	DAVID	WALLACE
MAFI112	FINAL	2023-01-30	10:00:00	12:00:00	1003	MICHEAL	SCOTT
MAMID111	MIDTERM	2023-01-05	20:00:00	21:30:00	1001	DAVID	WALLACE
OSFI141	FINAL	2023-01-30	10:00:00	12:00:00	1002	JAN	LEVISON
OSMID141	MIDTERM	2023-01-11	20:00:00	21:30:00	1003	MICHEAL	SCOTT

```

8 rows in set (0.001 sec)

```

- This command shows all of exams and the respective courses that are offering them.

SELECT

**Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STI
ME,Exam.EXAM_FTIME,
Course.COURSE_ID, Course.COURSE_NAME
FROM Exam,Course
WHERE Exam.COURSE_ID = Course.COURSE_ID;**

```
MariaDB [PROJECT]> SELECT Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STIME,Exam.EXAM_FTIME,
-> Course.COURSE_ID, Course.COURSE_NAME
-> FROM Exam,Course
-> WHERE Exam.COURSE_ID = Course.COURSE_ID;
```

EXAM_ID	EXAM_TYPE	EXAM_DATE	EXAM_STIME	EXAM_FTIME	COURSE_ID	COURSE_NAME
CPFI121	FINAL	2023-02-06	10:00:00	12:00:00	TCP1121	Computer Programming
CPMID121	LAB	2023-01-12	20:00:00	21:30:00	TCP1121	Computer Programming
DBFI131	FINAL	2023-02-15	10:00:00	12:00:00	TDB1131	Database Systems
DBMID131	MIDTERM	2023-01-03	20:00:00	21:30:00	TDB1131	Database Systems
MAFI112	FINAL	2023-01-30	10:00:00	12:00:00	TMA1111	Mathematical Techniques
MAMID111	MIDTERM	2023-01-05	20:00:00	21:30:00	TMA1111	Mathematical Techniques
OSFI141	FINAL	2023-01-30	10:00:00	12:00:00	TDB1131	Database Systems
OSMID141	MIDTERM	2023-01-11	20:00:00	21:30:00	TDB1131	Database Systems

8 rows in set (0.001 sec)

- To show all of the exams and the respective rooms that they will be held in.

SELECT

**Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STI
ME,Exam.EXAM_FTIME,
Room.ROOM_NUM, Room.ROOM_TYPE
FROM Exam,Room
WHERE Exam.ROOM_NUM = Room.ROOM_NUM;**

```
MariaDB [PROJECT]> SELECT Exam.EXAM_ID,Exam.EXAM_TYPE,Exam.EXAM_DATE,Exam.EXAM_STIME,Exam.EXAM_FTIME,
-> Room.ROOM_NUM, Room.ROOM_TYPE
-> FROM Exam,Room
-> WHERE Exam.ROOM_NUM = Room.ROOM_NUM;
```

EXAM_ID	EXAM_TYPE	EXAM_DATE	EXAM_STIME	EXAM_FTIME	ROOM_NUM	ROOM_TYPE
CPFI121	FINAL	2023-02-06	10:00:00	12:00:00	102	EXAM-ROOM
CPMID121	LAB	2023-01-12	20:00:00	21:30:00	101	LAB
DBFI131	FINAL	2023-02-15	10:00:00	12:00:00	104	EXAM-ROOM
DBMID131	MIDTERM	2023-01-03	20:00:00	21:30:00	102	EXAM-ROOM
MAFI112	FINAL	2023-01-30	10:00:00	12:00:00	102	EXAM-ROOM
MAMID111	MIDTERM	2023-01-05	20:00:00	21:30:00	104	EXAM-ROOM
OSFI141	FINAL	2023-01-30	10:00:00	12:00:00	102	EXAM-ROOM
OSMID141	MIDTERM	2023-01-11	20:00:00	21:30:00	104	EXAM-ROOM

8 rows in set (0.001 sec)

**The screenshots of the following three commands are all in one screenshot.*

- To show the attendance of all the students in course TCP1121.

```
SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE  
FROM Enroll  
WHERE COURSE_ID = 'TCP1121';
```

- To show courses and attendance of the student with ID number 1208.

```
SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE  
FROM Enroll  
WHERE STU_ID= 1208;
```

- To show the courses, attendance and status of the student with student ID 1205.

```
SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE,  
Enroll.STATUS  
FROM Enroll  
WHERE STU_ID= 1205;
```



```

MariaDB [PROJECT]> SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE
-> FROM Enroll
-> WHERE COURSE_ID = 'TCP1121';
+-----+-----+-----+
| COURSE_ID | STU_ID | STU_ATTENDANCE |
+-----+-----+-----+
| TCP1121   | 1200   | 80.00          |
| TCP1121   | 1201   | 89.90          |
| TCP1121   | 1203   | 84.00          |
| TCP1121   | 1205   | 88.00          |
| TCP1121   | 1208   | 81.00          |
+-----+-----+-----+
5 rows in set (0.021 sec)

MariaDB [PROJECT]>
MariaDB [PROJECT]>
MariaDB [PROJECT]> SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE
-> FROM Enroll
-> WHERE STU_ID= 1208;
+-----+-----+-----+
| COURSE_ID | STU_ID | STU_ATTENDANCE |
+-----+-----+-----+
| TCP1121   | 1208   | 81.00          |
| TDB1131   | 1208   | 66.67          |
| TMA1111   | 1208   | 88.00          |
| TOS1141   | 1208   | 80.00          |
+-----+-----+-----+
4 rows in set (0.001 sec)

MariaDB [PROJECT]>
MariaDB [PROJECT]>
MariaDB [PROJECT]> SELECT Enroll.COURSE_ID,Enroll.STU_ID, Enroll.STU_ATTENDANCE, Enroll.STATUS
-> FROM Enroll
-> WHERE STU_ID= 1205;
+-----+-----+-----+-----+
| COURSE_ID | STU_ID | STU_ATTENDANCE | STATUS |
+-----+-----+-----+-----+
| TCP1121   | 1205   | 88.00          | ELIGIBLE |
| TDB1131   | 1205   | 94.00          | ELIGIBLE |
| TMA1111   | 1205   | 45.00          | NULL    |
| TOS1141   | 1205   | 99.99          | ELIGIBLE |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)

```

- To insert new rooms using a trigger function.

```
CREATE TABLE TRI_TEST (  
  NEW_UPT VARCHAR(30));
```

```
DELIMITER $$
```

```
CREATE
```

```
  TRIGGER my_trigger BEFORE INSERT
```

```
  ON Room
```

```
  FOR EACH ROW BEGIN
```

```
    INSERT INTO TRI_TEST (NEW_UPT) VALUES('NEW ROOM  
    ADDED');
```

```
  END$$
```

```
DELIMITER ;
```

```
INSERT INTO Room VALUES  
(106,'LAB');
```

```
SELECT * FROM Room;
```

```
  SELECT * FROM TRI_TEST
```

```

XAMPP for Windows - mysql -u root
MariaDB [PROJECT]> CREATE TABLE TRI_TEST(
  -> NEW_UPT VARCHAR(30));
Query OK, 0 rows affected (0.024 sec)

MariaDB [PROJECT]> DELIMITER $$
MariaDB [PROJECT]>
MariaDB [PROJECT]> CREATE
  -> TRIGGER my_trigger BEFORE INSERT
  -> ON Room
  -> FOR EACH ROW BEGIN
  -> INSERT INTO TRI_TEST (NEW_UPT) VALUES('NEW ROOM ADDED');
  -> END$$
Query OK, 0 rows affected (0.031 sec)

MariaDB [PROJECT]> DELIMITER ;
MariaDB [PROJECT]>
MariaDB [PROJECT]> INSERT INTO Room VALUES
  -> (106,'LAB');
Query OK, 1 row affected (0.005 sec)

MariaDB [PROJECT]>
MariaDB [PROJECT]> SELECT * FROM Room;
+-----+-----+
| ROOM_NUM | ROOM_TYPE |
+-----+-----+
| 101 | LAB |
| 102 | EXAM-ROOM |
| 104 | EXAM-ROOM |
| 106 | LAB |
+-----+-----+
4 rows in set (0.001 sec)

MariaDB [PROJECT]> SELECT * FROM TRI_TEST;
+-----+
| NEW_UPT |
+-----+
| NEW ROOM ADDED |
+-----+
1 row in set (0.001 sec)

```

- To create a flag for students with low attendance but pardon them to sit for exams using procedure.

delimiter &&

CREATE PROCEDURE attendance_pardon(IN courseId varchar(20), IN id int(10))

BEGIN

UPDATE enroll

SET STU_Attendance='99.99'

WHERE COURSE_ID = courseId

AND STU_ID = id;

UPDATE enroll

SET STATUS = 'ELIGIBLE'

WHERE COURSE_ID = courseId

AND STU_ID = id;

END &&

delimiter ;

SELECT * FROM enroll WHERE STU_ID='1201';

CALL attendance_pardon('TDB1131',1201);

SELECT * FROM enroll WHERE STU_ID='1201';

```
MariaDB [PROJECT]> CREATE PROCEDURE attendance_pardon(IN courseId varchar(20), IN id int(10))
-> BEGIN
-> UPDATE enroll
-> SET STU_Attendance='99.99'
-> WHERE COURSE_ID = courseId
-> AND STU_ID = id;
-> UPDATE enroll
-> SET STATUS = 'ELIGIBLE'
-> WHERE COURSE_ID = courseId
-> AND STU_ID = id;
-> END &&
Query OK, 0 rows affected (0.008 sec)

MariaDB [PROJECT]> delimiter ;
```

```
MariaDB [PROJECT]> SELECT * FROM enroll WHERE STU_ID='1201';
```

COURSE_ID	STU_ID	STU_ATTENDANCE	STATUS
TCP1121	1201	89.90	ELIGIBLE
TDB1131	1201	67.00	NULL
TMA1111	1201	83.11	ELIGIBLE
TOS1141	1201	78.65	NULL

```
4 rows in set (0.001 sec)
```

```
MariaDB [PROJECT]> call attendance_pardon('TDB1131',1201)
-> ;
Query OK, 2 rows affected (0.007 sec)
```

```
MariaDB [PROJECT]> SELECT * FROM enroll WHERE STU_ID='1201';
```

COURSE_ID	STU_ID	STU_ATTENDANCE	STATUS
TCP1121	1201	89.90	ELIGIBLE
TDB1131	1201	99.99	ELIGIBLE
TMA1111	1201	83.11	ELIGIBLE
TOS1141	1201	78.65	NULL

```
4 rows in set (0.001 sec)
```

- To format new entries of invigilator's name uppercase

delimiter &&

```
CREATE TRIGGER invg_name  
BEFORE INSERT ON invigilator  
FOR EACH ROW  
BEGIN  
SET NEW.INVIGILATOR_LNAME=  
UPPER(NEW.INVIGILATOR_LNAME);  
SET NEW.INVIGILATOR_FNAME=  
UPPER(NEW.INVIGILATOR_FNAME);  
END&&  
delimiter ;
```

insert into invigilator values('0000', 'dummy', 'dummy');

select * from invigilator;

```
MariaDB [PROJECT]> delimiter &&  
MariaDB [PROJECT]>  
MariaDB [PROJECT]> CREATE TRIGGER invg_name  
-> BEFORE INSERT ON invigilator  
-> FOR EACH ROW  
-> BEGIN  
-> SET NEW.INVIGILATOR_LNAME= UPPER(NEW.INVIGILATOR_LNAME);  
-> SET NEW.INVIGILATOR_FNAME= UPPER(NEW.INVIGILATOR_FNAME);  
-> END&&  
Query OK, 0 rows affected (0.013 sec)
```

```
MariaDB [PROJECT]> insert into invigilator values  
-> ('0000', 'dummy', 'dummy');  
Query OK, 1 row affected (0.006 sec)
```

```
MariaDB [PROJECT]> select * from invigilator;
```

INVIGILATOR_ID	INVIGILATOR_FNAME	INVIGILATOR_LNAME
0	DUMMY	DUMMY
1001	DAVID	WALLACE
1002	JAN	LEVISON
1003	MICHEAL	SCOTT
1004	ANGELA	MARTIN
1005	STANLEY	HUDSON

```
6 rows in set (0.001 sec)
```

- To display the total number of students per course.

```
SELECT COURSE_ID, COUNT(STU_ID) FROM ENROLL GROUP BY COURSE_ID;
```

```
SELECT COURSE.COURSE_NAME, COUNT(ENROLL.STU_ID) as count_student  
FROM ENROLL
```

```
JOIN COURSE ON COURSE.COURSE_ID = ENROLL.COURSE_ID  
GROUP BY ENROLL.COURSE_ID, COURSE.COURSE_NAME;
```

```

MariaDB [testingdatabase]> SELECT COURSE_ID, COUNT(STU_ID) FROM ENROLL GROUP BY COURSE_ID;
+-----+-----+
| COURSE_ID | COUNT(STU_ID) |
+-----+-----+
| TCP1121   | 5              |
| TDB1131   | 10             |
| TMA1111   | 8              |
| TOS1141   | 7              |
+-----+-----+
4 rows in set (0.002 sec)

MariaDB [testingdatabase]> SELECT COURSE.COURSE_NAME, COUNT(ENROLL.STU_ID) as count_student
-> FROM ENROLL
-> JOIN COURSE ON COURSE.COURSE_ID = ENROLL.COURSE_ID
-> GROUP BY ENROLL.COURSE_ID, COURSE.COURSE_NAME;
+-----+-----+
| COURSE_NAME | count_student |
+-----+-----+
| Computer Programming | 5              |
| Database Systems    | 10             |
| Mathematical Techniques | 8              |
| Operating Systems   | 7              |
+-----+-----+
4 rows in set (0.002 sec)

```


9 Business Ideas

Some business ideas that our group have come up with is a subscription-based model where we can charge educational institutions or students a recurring fee for access to the software. We can also create Pay-per-use model whereby we may charge educational institutions or students for each exam scheduled through the software.

In addition to asking customers to pay us, this project may also make a move towards advertising whereby we partner with educational institutions or businesses and sell advertising space within the software. This will increase the brands recognition and eventually this database system can integrate with other systems. By integrating the exam scheduling software with other systems such as learning management systems, student information systems, or financial aid systems, we can charge for the integration.

Furthermore, the database system may also provide premium features and additional features such as exam proctoring, analytics, or accessibility support for a higher, monthly or yearly recurring fee. Another premium feature that can be offered by the data base system is Data analytics. The system can offer data analytics services and insights based on the data collected from the exam scheduling software. Furthermore. The system can also offer additional customization. By offering customization services to educational institutions, the group can charge extra for the customization of the software to their specific needs.

10 Advantages and disadvantages

Ease of use:

A basic system may have a simple and user-friendly interface, making it easy for users to navigate and use effectively.

Low cost:

A basic system may be less expensive than more advanced systems, making it more accessible to smaller institutions or organizations.

Quick implementation:

A basic system may have a shorter implementation time, allowing institutions to quickly begin scheduling exams.

Basic functionalities:

A basic system may have the basic functionalities that are needed for scheduling exams and resolving conflicts.

Suitable for small scale:

A basic system may be suitable for small institutions or organizations that do not need advanced features or scalability.

Limited functionality:

A basic system may only have basic features such as exam scheduling and conflict resolution and may not have advanced features such as exam proctoring, analytics, or accessibility support.

Lack of scalability:

A basic system may not be able to handle a large number of exams or students and may not be able to scale to meet the needs of a growing institution.

Limited integration:

A basic system may not be able to integrate with other systems such as learning management systems, student information systems, or financial aid systems, which can limit its usefulness.