Activity No. 3	
Primary and Foreign Keys	
Course Code: CPE011	**Program:** BSCPE
Course Title: Database Management System	**Date Performed:** September 2, 2022
Section: CPE21S3	**Date Submitted:** September 2, 2022
Name: Aaron Martin Parallag Caro Christian Efa	**Instructor:** Dr. Jonathan Vidal Taylar

1. Objective(s):

This activity aims to discuss the purpose of database tables with the Primary Key Constraint, the creation of databases and alteration of the existing structure to utilize the primary key functions.

2. Intended Learning Outcomes (ILOs):

The students should be able to: 2.1 Understand and discuss the purpose of primary and foreign keys in a database structure. 2.2 Create or Alter a database table to include Primary and Foreign Keys.

3. Discussion:

In the previous activity, we have seen that our initial database design did not account for maintaining the uniqueness of each row so that we were able to insert an exact duplicate of one row to a new row and we saw that deleting one of them results in the deletion of the redundant tables which means data could be lost permanently. This can be solved by using a primary key.

Primary Keys

A primary key is a column or a set of columns that uniquely identifies each row in the table. You must follow the rules below when you define a primary key for a table:

- A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot contain NULL values. It means that you have to declare the primary key column with the NOT NULL attribute. If you don't, MySQL will force the primary key column as NOT NULL implicitly.
- A table has only one primary key. Also, a primary key column often has the AUTO_INCREMENT attribute that generates a unique sequence for the key automatically. The primary key of the next row is greater than the previous one.

To insert a primary key, one can add the constraint during table creation. To do this, the following syntax can be used.

```
CREATE TABLE Persons
(
    P_Id int NOT NULL,
    LastNamevarchar(255) NOT NULL,
    FirstNamevarchar(255),
    Address varchar(255),
    City varchar(255),
    PRIMARY KEY (P_Id)
)
```

Note that P_Id is the primary key. Another method in adding a primary key is using the alter table command. The following syntax can be used.

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id);
```

To drop a PRIMARY KEY constraint, use the following SQL Command:

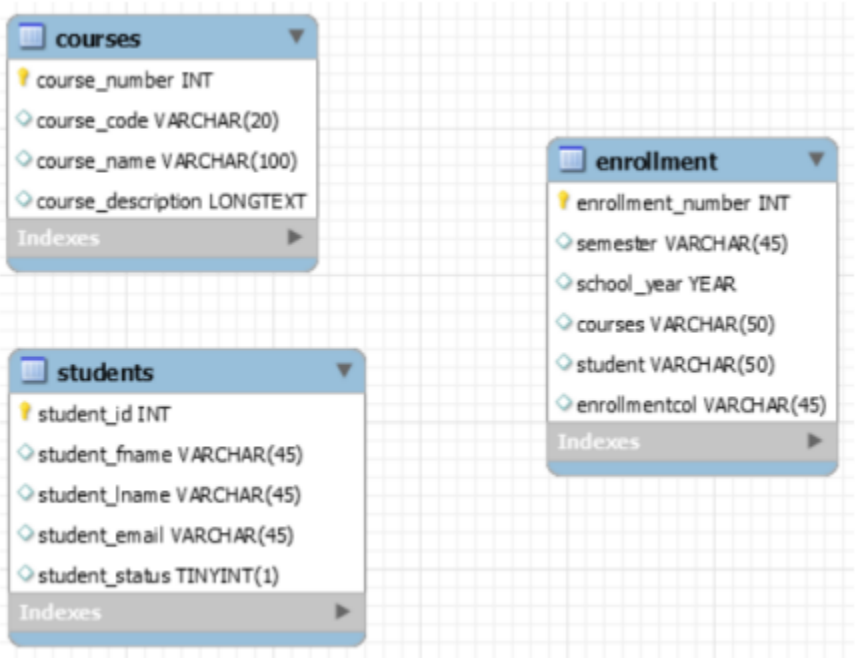
ALTER TABLE Persons

DROP PRIMARY KEY

Foreign Keys

A foreign key is a field in a table that matches/links another field of another table. A foreign key places constraints on data in the related tables, which enables MySQL to maintain referential integrity.

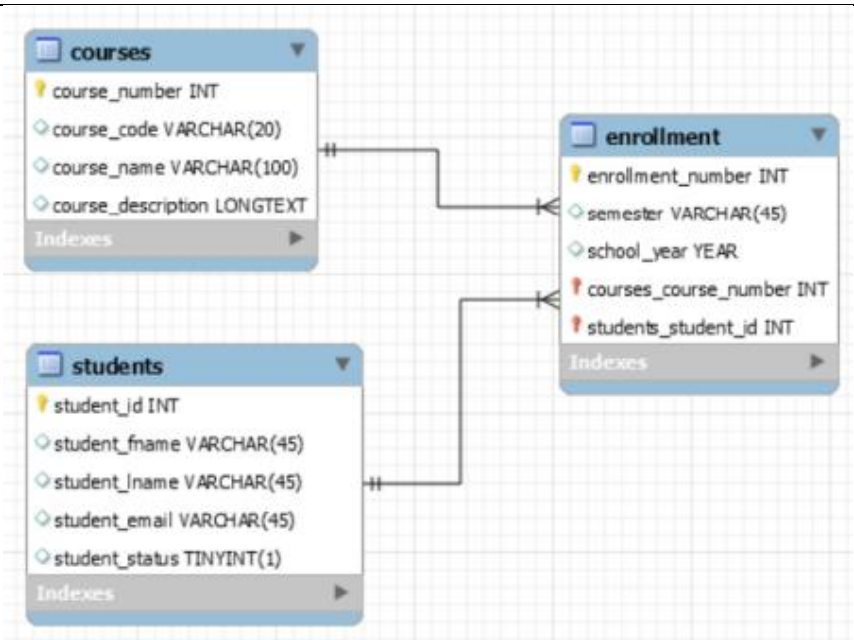
Consider the following database diagram:



Observe that each table now has its own Primary Key which eliminates redundant records in that table and keeps each record unique, If we populate each table we will have no redundant records as the PRIMARY KEY constraint will prevent this.

A new problem can be observed when we take a look at the enrollment table. It has a student name and course code. If we tried inserting a record that does not in either students or courses it will still be accepted. You may try implementing the schema in your database. The problem encountered with this type of database design is a violation of the Referential Integrity constraint. The accuracy and the consistency of data cannot be validated with this database design.

Adding a Foreign Key constraint resolves this problem. We will build a relationship – connect enrollment with students. There are three types of relationship between tables that they can have: One to one, One to Many, and Many to Many. MySQL does not have an option to specify this. It depends on your implementation of the primary keys and foreign keys.



We have now established a relationship between the two tables through foreign keys (denoted by the red bulbs) on the enrollment table. MySQL will now deny any insertion of data on the course and student field that is not existing in courses and students.

To create a foreign key constraint during table creation process, note that the besides the key itself, a constraint is also created. The syntax for creating a foreign key follows this syntax:

FOREIGN KEY constraint_name(column)
REFERENCES reference_table(reference_pkColumn)

Where constraint name is the name of the foreign key constraint that was added to that column. Make sure to always put a constraint name otherwise MySQL will automatically generate one.

```
CREATE TABLE enrollment
 (
        enrollment_number INT,
        student INT, course INT,
        semester VARCHAR(20),
        PRIMARY KEY(enrollment_number),
        FOREIGN KEY fk_student(student) REFERENCES students(student_id,)
        FOREIGN KEY fk_course(course) REFERENCES courses(course_number)
 );
```

Observe that we need to create the fields first before assigning them the Primary Key and Foreign Key constraint.

To create a FOREIGN KEY constraint when the enrollment table or when you've already created a table. You can use the following command:

ALTER TABLE enrollment
ADD FOREIGN KEY fk_student(student) REFERENCES students(student_id);

Adding Foreign keys using the ALTER command is one at a time. You cannot add multiple foreign keys in one statement. To drop a foreign key, use the following command syntax:

ALTER TABLE enrollment DROP FOREIGN KEY constraint_name;
ALTER TABLE enrollment DROP KEY constraint_name;

You need to first drop the constraint/rule foreign key then drop the actual key/index.

4. Materials and Equipment:

Desktop Computer
Windows Operating System
XAMPP Application

5. Procedure:

1. Create a database studentdb_;
2. Create a table named students with attributes:
   - ID(Primary Key)
   - First Name
   - Last Name
   - Program
3. Create another table named Courses with attributes
   - Course Code(Primary Key)
   - Course Description
   - Department(Not Null)
4. Create a third table named Sections with Attributes
   - Student Number(References from students)
   - Course Code(References from courses)
   - Section (Primary Key)
   - Class Code 5. Create a fourth table named classrooms with attributes
   - Room Number (3210, 5204)
   - Time (ex. 730, 830, 1230, 1330)
   - Duration
   - Course Code (Reference from courses)
   - Section (Reference from sections)
6. Place the screenshot of the table structure of each table in Section 6.
7. Try inserting a value into classrooms. What is the result? _____
8. Try inserting a value into sections. What is the result? _____
9. Populate the students and courses table with 5 records each.
10. Take screenshots of your table values for each table write your observation below the image.
11. Insert records into sections with a student number based from existing student IDs and with a course code based from existing course codes records. What is the result? _____
12. Insert records into sections with a student number that does not exist student table ID but has a course code that exists in the courses table. What is the result? _____
13. Populate the sections table with at least 5 records.
14. Take a screenshot of the values in sections table. Write your observations below the image.
15. Insert values into classrooms with the same records (ex. 3210,730,2,'CPE011','CPE21FB1') two times. What was the result? _____
16. Delete all the values in classroom.
17. Implement a primary key in classrooms. Use techniques you've learned previously to add a classroom ID.
18. Try inserting two identical records into classroom again. What is the result? _____
19. Delete all the values in classroom.
20. Run this command: ALTER TABLE classrooms ADD UNIQUE KEY uk_roomtime (room_number,time);
21. Try inserting two identical records into classrooms again. What is the result? _____
22. Populate classroom with at least 5 records.
23. Take screenshots of the values in classrooms. Write observations below the image.

6. Database Output:

Copy screenshot(s) of your output with observations after completing the procedures provided in Part 5.

1. Create a database studentdb_;

```
MariaDB [(none)]> CREATE database studentsdb_Caro_Efa;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use studentsdb_Caro_Efa;
Database changed
MariaDB [studentsdb_Caro_Efa]>
```

2. Create a table named students with attributes:
   - ID(Primary Key)
   - First Name
   - Last Name
   - Program

```
MariaDB [studentsdb_Caro_Efa]> CREATE TABLE STUDENTS(
    -> ID INT AUTO_INCREMENT PRIMARY KEY,
    -> FIRST_NAME VARCHAR(100),
    -> LAST_NAME VARCHAR(100),
    -> PROGRAM VARCHAR(20));
Query OK, 0 rows affected (0.020 sec)
```

3. Create another table named Courses with attributes
   - Course Code(Primary Key)
   - Course Description
   - Department(Not Null)

```
MariaDB [studentsdb_Caro_Efa]> CREATE TABLE COURSES(
    -> COURSE_CODE INT AUTO_INCREMENT PRIMARY KEY,
    -> COURSE_DESCRIPTION VARCHAR(255),
    -> DEPARTMENT VARCHAR(20) NOT NULL);
Query OK, 0 rows affected (0.019 sec)
```

4. Create a third table named Sections with Attributes
   - Student Number(References from students)
   - Course Code(References from courses)
   - Section (Primary Key)
   - Class Code

```
MariaDB [studentsdb_Caro_Efa]> CREATE TABLE SECTIONS(
    -> STUDENT_NUMBER INT,
    -> COURSE_CODE INT,
    -> SECTION VARCHAR(100) PRIMARY KEY,
    -> CLASS_CODE VARCHAR(255),
    -> FOREIGN KEY (STUDENT_NUMBER) REFERENCES STUDENTS(ID),
    -> FOREIGN KEY(COURSE_CODE) REFERENCES COURSES(COURSE_CODE)
    -> );
Query OK, 0 rows affected (0.023 sec)
```

5. Create a fourth table named classrooms with attributes
- Room Number (3210, 5204)
- Time (ex. 730, 830, 1230, 1330)
- Duration
- Course Code (Reference from courses)
- Section (Reference from sections)

```
MariaDB [studentsdb_Caro_Efa]> CREATE TABLE CLASSROOMS(
    -> ROOM_NUMBER INT,
    -> TIME INT,
    -> DURATION VARCHAR(100),
    -> COURSE_CODE INT,
    -> SECTION VARCHAR(100),
    -> FOREIGN KEY (COURSE_CODE) REFERENCES COURSES(COURSE_CODE),
    -> FOREIGN KEY (SECTION) REFERENCES SECTIONS(SECTION)
    -> );
Query OK, 0 rows affected (0.027 sec)
```

6. Place the screenshot of the table structure of each table in Section 6.

```
MariaDB [studentsdb_Caro_Efa]> DESC STUDENTS;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| ID         | int(11)      | NO   | PRI | NULL    | auto_increment |
| FIRST_NAME | varchar(100) | YES  |     | NULL    |                |
| LAST_NAME  | varchar(100) | YES  |     | NULL    |                |
| PROGRAM    | varchar(20)  | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.011 sec)

MariaDB [studentsdb_Caro_Efa]> DESC COURSES;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| COURSE_CODE        | int(11)      | NO   | PRI | NULL    | auto_increment |
| COURSE_DESCRIPTION | varchar(255) | YES  |     | NULL    |                |
| DEPARTMENT         | varchar(20)  | NO   |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
3 rows in set (0.008 sec)

MariaDB [studentsdb_Caro_Efa]> DESC SECTIONS;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| STUDENT_NUMBER | int(11)      | YES  | MUL | NULL    |       |
| COURSE_CODE    | int(11)      | YES  | MUL | NULL    |       |
| SECTION        | varchar(100) | NO   | PRI | NULL    |       |
| CLASS_CODE     | varchar(255) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
4 rows in set (0.008 sec)

MariaDB [studentsdb_Caro_Efa]> DESC CLASSROOMS;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| ROOM_NUMBER | int(11)      | YES  |     | NULL    |       |
| TIME        | int(11)      | YES  |     | NULL    |       |
| DURATION    | varchar(100) | YES  |     | NULL    |       |
| COURSE_CODE | int(11)      | YES  | MUL | NULL    |       |
| SECTION     | varchar(100) | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.008 sec)
```

7. Try inserting a value into classrooms. What is the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (8,7,'120 MINUTES',5,'S3');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`studentsdb_caro_efa`.`classrooms`, CONSTRAINT
`classrooms_ibfk_1` FOREIGN KEY (`COURSE_CODE`) REFERENCES `courses` (`COURSE_CODE`))
MariaDB [studentsdb_Caro_Efa]>
```

8. Try inserting a value into sections. What is the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (51234,5,'S3','234');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`studentsdb_caro_efa`.`sections`, CONSTRAINT `s
ections_ibfk_1` FOREIGN KEY (`STUDENT_NUMBER`) REFERENCES `students` (`ID`))
MariaDB [studentsdb_Caro_Efa]>
```

9. Populate the students and courses table with 5 records each.

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO STUDENTS (FIRST_NAME, LAST_NAME, PROGRAM)
    -> VALUES('ken', 'chana','IT');
Query OK, 1 row affected (0.006 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO STUDENTS (FIRST_NAME, LAST_NAME, PROGRAM)
    -> VALUES('ken2', 'chana2','IT2');
Query OK, 1 row affected (0.002 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO STUDENTS (FIRST_NAME, LAST_NAME, PROGRAM)
    -> VALUES('ken3', 'chana3','IT3');
Query OK, 1 row affected (0.004 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO STUDENTS (FIRST_NAME, LAST_NAME, PROGRAM)
    -> VALUES('ken4', 'chana4','IT4');
Query OK, 1 row affected (0.002 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO STUDENTS (FIRST_NAME, LAST_NAME, PROGRAM)
    -> VALUES('ken5', 'chana5','IT5');
Query OK, 1 row affected (0.001 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO COURSES (COURSE_CODE, COURSE_DESCRIPTION, DEPARTMENT)
    -> VALUES (5,'IT','PROG');
Query OK, 1 row affected (0.004 sec)

MariaDB [studentsdb_Caro_Efa]>  INSERT INTO COURSES (COURSE_CODE, COURSE_DESCRIPTION, DEPARTMENT)
    -> VALUES (52,'IT2','PROG2');
Query OK, 1 row affected (0.005 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO COURSES (COURSE_CODE, COURSE_DESCRIPTION, DEPARTMENT)
    -> VALUES (53,'IT3','PROG3');
Query OK, 1 row affected (0.004 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO COURSES (COURSE_CODE, COURSE_DESCRIPTION, DEPARTMENT)
    -> VALUES (54,'IT4','PROG4');
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO COURSES (COURSE_CODE, COURSE_DESCRIPTION, DEPARTMENT)
    -> VALUES (55,'IT5','PROG5');
Query OK, 1 row affected (0.002 sec)

MariaDB [studentsdb_Caro_Efa]>
```

10. Take screenshots of your table values for each table write your observation below the image.

```
MariaDB [studentsdb_Caro_Efa]> SELECT * FROM STUDENTS;
+----+------------+-----------+---------+
| ID | FIRST_NAME | LAST_NAME | PROGRAM |
+----+------------+-----------+---------+
|  1 | ken        | chana     | IT      |
|  2 | ken2       | chana2    | IT2     |
|  3 | ken3       | chana3    | IT3     |
|  4 | ken4       | chana4    | IT4     |
|  5 | ken5       | chana5    | IT5     |
+----+------------+-----------+---------+
5 rows in set (0.002 sec)

MariaDB [studentsdb_Caro_Efa]> SELECT * FROM COURSES;
+-------------+--------------------+------------+
| COURSE_CODE | COURSE_DESCRIPTION | DEPARTMENT |
+-------------+--------------------+------------+
|           5 | IT                 | PROG       |
|          52 | IT2                | PROG2      |
|          53 | IT3                | PROG3      |
|          54 | IT4                | PROG4      |
|          55 | IT5                | PROG5      |
+-------------+--------------------+------------+
5 rows in set (0.000 sec)

MariaDB [studentsdb_Caro_Efa]>
```

11. Insert records into sections with a student number based from existing student IDs and with a course code based from existing course codes records. What is the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (1,5,'S3',24);
Query OK, 1 row affected (0.001 sec)

MariaDB [studentsdb_Caro_Efa]>
```

12. Insert records into sections with a student number that does not exist student table ID but has a course code that exists in the courses table. What is the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (6,5,'S4',24);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`studentsdb_caro_efa`.`sections`, CONSTRAINT `s
ections_ibfk_1` FOREIGN KEY (`STUDENT_NUMBER`) REFERENCES `students` (`ID`))
MariaDB [studentsdb_Caro_Efa]>
```

13. Populate the sections table with at least 5 records.

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (2,52,'S4',25);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (3,53,'S5',26);
Query OK, 1 row affected (0.004 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (4,54,'S6',27);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (5,55,'S7',28);
Query OK, 1 row affected (0.005 sec)

MariaDB [studentsdb_Caro_Efa]>
```

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO SECTIONS VALUES (1,5,'S3',24);
Query OK, 1 row affected (0.001 sec)

MariaDB [studentsdb_Caro_Efa]>
```

14. Take a screenshot of the values in sections table. Write your observations below the image.

```
MariaDB [studentsdb_Caro_Efa]> SELECT * FROM SECTIONS;
+----------------+-------------+---------+------------+
| STUDENT_NUMBER | COURSE_CODE | SECTION | CLASS_CODE |
+----------------+-------------+---------+------------+
|              1 |           5 | S3      | 24         |
|              2 |          52 | S4      | 25         |
|              3 |          53 | S5      | 26         |
|              4 |          54 | S6      | 27         |
|              5 |          55 | S7      | 28         |
+----------------+-------------+---------+------------+
5 rows in set (0.000 sec)

MariaDB [studentsdb_Caro_Efa]>
```

15. Insert values into classrooms with the same records (ex. 3210,730,2,'CPE011','CPE21FB1') two times. What was the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUE(3210,730,2,'5','S3');
Query OK, 1 row affected (0.002 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUE(3210,730,2,'5','S3');
Query OK, 1 row affected (0.001 sec)

MariaDB [studentsdb_Caro_Efa]>
```

16. Delete all the values in classroom.

```
MariaDB [studentsdb_Caro_Efa]> DELETE FROM CLASSROOMS WHERE TIME =730;
Query OK, 2 rows affected (0.005 sec)

MariaDB [studentsdb_Caro_Efa]>
```

17. Implement a primary key in classrooms. Use techniques you've learned previously to add a classroom ID.

```
MariaDB [studentsdb_Caro_Efa]> ALTER TABLE CLASSROOMS ADD CLASSROOM_ID VARCHAR(255);
Query OK, 0 rows affected (0.010 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [studentsdb_Caro_Efa]> ALTER TABLE CLASSROOMS ADD PRIMARY KEY (CLASSROOM_ID);
Query OK, 0 rows affected (0.036 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

18. Try inserting two identical records into classroom again. What is the result?

```
ERROR 1136 (21S01): Column count doesn't match value count at row 1
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (3210,730,2,'5','S3',11)
    -> ;
Query OK, 1 row affected (0.002 sec)

MariaDB [studentsdb_Caro_Efa]>  INSERT INTO CLASSROOMS VALUES (3210,730,2,'5','S3',11)
    -> ;
ERROR 1062 (23000): Duplicate entry '11' for key 'PRIMARY'
MariaDB [studentsdb_Caro_Efa]>
```

19. Delete all the values in classroom.

```
MariaDB [studentsdb_Caro_Efa]> DELETE FROM CLASSROOMS WHERE COURSE_CODE = 5
    -> ;
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]>
```

20. Run this command: ALTER TABLE classrooms ADD UNIQUE KEY uk_roomtime (room_number,time);

```
MariaDB [studentsdb_Caro_Efa]> ALTER TABLE CLASSROOMS ADD UNIQUE KEY UK_ROOMTIME (ROOM_NUMBER, TIME);
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [studentsdb_Caro_Efa]>
```

21. Try inserting two identical records into classrooms again. What is the result?

```
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (3210,730,2,5,'S3',11);
ERROR 1062 (23000): Duplicate entry '11' for key 'PRIMARY'
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (3210,730,2,5,'S3',11);
ERROR 1062 (23000): Duplicate entry '11' for key 'PRIMARY'
MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (3210,730,2,5,'S3',1);
ERROR 1062 (23000): Duplicate entry '3210-730' for key 'UK_ROOMTIME'
MariaDB [studentsdb_Caro_Efa]>
```

22. Populate classroom with at least 5 records.

```
MariaDB [studentsdb_Caro_Efa]>  INSERT INTO CLASSROOMS VALUES (321,230,3,52,'S4',2);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]>  INSERT INTO CLASSROOMS VALUES (421,330,4,53,'S5',3);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (521,430,5,54,'S6',4);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (621,530,6,55,'S7',5);
Query OK, 1 row affected (0.003 sec)

MariaDB [studentsdb_Caro_Efa]>

MariaDB [studentsdb_Caro_Efa]> INSERT INTO CLASSROOMS VALUES (3210,730,2,5,'S3',11)
    -> ;
Query OK, 1 row affected (0.003 sec)
```

23. Take screenshots of the values in classrooms. Write observations below the image.

```
MariaDB [studentsdb_Caro_Efa]> SELECT * FROM CLASSROOMS;
+-------------+------+----------+-------------+---------+--------------+
| ROOM_NUMBER | TIME | DURATION | COURSE_CODE | SECTION | CLASSROOM_ID |
+-------------+------+----------+-------------+---------+--------------+
|        3210 |  730 | 2        |           5 | S3      | 11           |
|         321 |  230 | 3        |          52 | S4      | 2            |
|         421 |  330 | 4        |          53 | S5      | 3            |
|         521 |  430 | 5        |          54 | S6      | 4            |
|         621 |  530 | 6        |          55 | S7      | 5            |
+-------------+------+----------+-------------+---------+--------------+
5 rows in set (0.000 sec)

MariaDB [studentsdb_Caro_Efa]>
```

7. Supplementary Activity:

Questions:

1. Explain the results of step 7 and 8 in Procedure. Why did they give the outputs that they did?
In step 7 and 8 we tried to insert values into every rows of table **classrooms** and **sections** it did not allow us to update the foreign key because of constraints, so we concluded that a foreign key cannot be updated unless we update the primary key in its respective table. **Note:** A primary key is a unique key of a certain table and there can only be one in each table.

2. Explain the results of step 11 and 12 in Procedure. Why did they give the outputs that they did?
In step 11 we just copied the values of the primary keys in table **students** and **course**s, the row **STUDENT_NUMBER** and **COURSE_CODE** are foreign keys which became a placeholder of the primary keys of **STUDENTS** and **COURSES**. Then we inserted values into rows **SECTION** and **CLASS_CODE.**

In step 12 it failed to accept the inputted value because it did not recognize it, instead of adding a value the database recognize the command as updating a value which cannot be done as explained in the first question.

3. Why is it necessary to implement a primary key and foreign key for a database?
Primary and foreign keys help create a linked structure or relational databases in the system. A primary key ensures unique row identification. This results in faster sorting, searching, and querying operations. This is illustrated in the discussion section of the activity

4. Cite your own real-life situation where a primary key and foreign key in a database is necessary. Explain.
In a scenario that we need to input values but I want it to be unique we would use a primary key and foreign key. For instance, we are sorting books and we have books that have the same title but we want to uniquely distinguish them so by using a primary key although they have the same title what separates them is their unique identification code. Let say we have three books of Narnia volume 1 having an identifier like the primary key would let me know which of the Narnia volume 1 we lend to the other person.

5. Why did implementing the primary key in classrooms fail to solve the redundancy problem?
Implementing the primary key did solve the redundancy problem as shown in section 6, at first without the primary key it accepted two inputs of the same value. Then the primary key was added

and tried to do the same thing imputing the same values twice, it accepted the first input and deny the second showing it resulted in a duplication error.

6. What do you think is the difference between the Primary Key and the Unique Key?

The main difference is that primary keys cannot accept null values but a unique key can accept null values, so imagine everyone has a national id that is a primary key but not everyone can have a voter's id because they are not of age and that is a unique key. Both primary keys and unique keys enforces unique values and as for unique key nothing can also be unique value, nothing is not equal to nothing in the case of unique key.

8. Conclusion:

We have learned that creating relations in databases requires uniqueness. With the use of primary keys and foreign keys we are able to establish a link to different tables remove redundancy and sort out data it an organized manner. From this activity we have realized the importance of constraints and relations to databases as it can hasten sorting, searching and even delivering queries to the user.

"We accept responsibility for our role in ensuring the integrity of the work submitted by the group in which we participated."