



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Project Report

CSU34041 Information Management II

5th December 2020

1 – Introduction.....	2
2 – Tables.....	2
3 – Views.....	3
4 – Triggers.....	4
5 – Diagrams.....	6
6 – Appendix.....	9

1 – Introduction

For my project, I designed a database that represents a primary school. This database could be used by the principle of the school to keep a record of the school and everything that goes on within. I implemented this database using mysql on the command line on my computer.

2 – Tables

2.1 – Students

The students table contained variables for each student's first name, last name, birth date, sex, class id and most importantly their student id which was the primary key. The foreign key was the class variable which pointed to the class id variable in the classes table

2.2 – Teachers

The teachers table contained variables for each teacher's first name, last name, birth date, sex, class id and teacher id which was the primary key. The foreign key was the class variable which pointed to the class id variable in the classes table.

2.3 – Absences

The absences table contained variables for each absence's student id, date and absence id which was the primary key. The foreign key was the student id which pointed to the student id variable in the students table.

2.4 – Classes

The classes table contained variables for each class' name, teacher id and class id which was the primary key. The foreign key was the teacher id which pointed to the teacher id variable in the teachers table.

2.5 – Scores

The scores table contained variables for each score's student id, score and score id which was the primary key for this table. The foreign key was the student id which pointed to the student id variable in the students table.

2.6 – Tests

The tests table contained variables for each test's date, type, class id and test id which was the primary key for this table. The foreign key was the class id which pointed to the class id variable in the classes table.

3 – Views

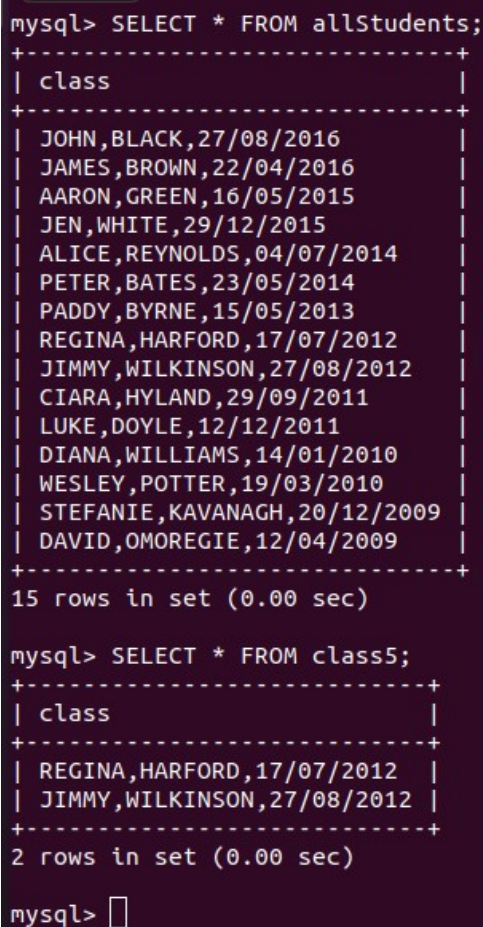
Views could be very useful when using a database like this. The principle of the school could use different views to view the students in his school. For example he could use this view which gives a list of all students names and dates of birth.

```
CREATE VIEW allStudents AS
  SELECT Concat(first_name, ',',last_name, ',',birth_date) AS class FROM students;
```

If we want to narrow a view down to a single class, we can do so and view all the students in a particular class like so.

```
CREATE VIEW class5 AS
  SELECT Concat(first_name, ',',last_name, ',',birth_date) AS class FROM students WHERE
  class_id = 5;
```

We can see this in the image below.



```
mysql> SELECT * FROM allStudents;
+-----+
| class |
+-----+
| JOHN,BLACK,27/08/2016 |
| JAMES,BROWN,22/04/2016 |
| AARON,GREEN,16/05/2015 |
| JEN,WHITE,29/12/2015 |
| ALICE,REYNOLDS,04/07/2014 |
| PETER,BATES,23/05/2014 |
| PADDY,BYRNE,15/05/2013 |
| REGINA,HARFORD,17/07/2012 |
| JIMMY,WILKINSON,27/08/2012 |
| CIARA,HYLAND,29/09/2011 |
| LUKE,DOYLE,12/12/2011 |
| DIANA,WILLIAMS,14/01/2010 |
| WESLEY,POTTER,19/03/2010 |
| STEFANIE,KAVANAGH,20/12/2009 |
| DAVID,OMOREGIE,12/04/2009 |
+-----+
15 rows in set (0.00 sec)

mysql> SELECT * FROM class5;
+-----+
| class |
+-----+
| REGINA,HARFORD,17/07/2012 |
| JIMMY,WILKINSON,27/08/2012 |
+-----+
2 rows in set (0.00 sec)

mysql> □
```

4 – Triggers

Triggers can be very useful when using a database like this. For example, I created a trigger that would change the first name of a student to upper case when it was inserted into the students table.

```
CREATE TRIGGER upperName
BEFORE INSERT ON students
FOR EACH ROW
SET NEW.first_name = UPPER(NEW.first_name);
```

I also created the trigger for the last names too.

```
CREATE TRIGGER upperLastName
BEFORE INSERT ON students
FOR EACH ROW
SET NEW.last_name = UPPER(NEW.last_name);
```

A slightly more complicated trigger that I created would update the absences table to remove all corresponding entries of a student if that student was deleted from the students table. I ran into trouble when typing this into the command line so I had to switch the delimiter for it to work, but I changed it back to default right after.

```
DELIMITER //
CREATE TRIGGER updateStudentAbsences
AFTER DELETE ON students
FOR EACH ROW
BEGIN
DELETE FROM absences
WHERE absences.student_id = old.student_id;
END//
DELIMITER ;
```

We can see this working in the screenshot below. After the delete is performed on the student, the absence is also deleted from the absences table.

```
mysql> SELECT * FROM absences;
+-----+-----+-----+
| student_id | date       | absence_id |
+-----+-----+-----+
| 1 | 15/09/2020 | 1 |
| 2 | 22/11/2020 | 2 |
| 3 | 07/12/2020 | 3 |
| 4 | 30/09/2020 | 4 |
| 5 | 28/09/2020 | 5 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> DELETE FROM students WHERE student_id =1;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM absences;
+-----+-----+-----+
| student_id | date       | absence_id |
+-----+-----+-----+
| 2 | 22/11/2020 | 2 |
| 3 | 07/12/2020 | 3 |
| 4 | 30/09/2020 | 4 |
| 5 | 28/09/2020 | 5 |
+-----+-----+-----+
4 rows in set (0.00 sec)

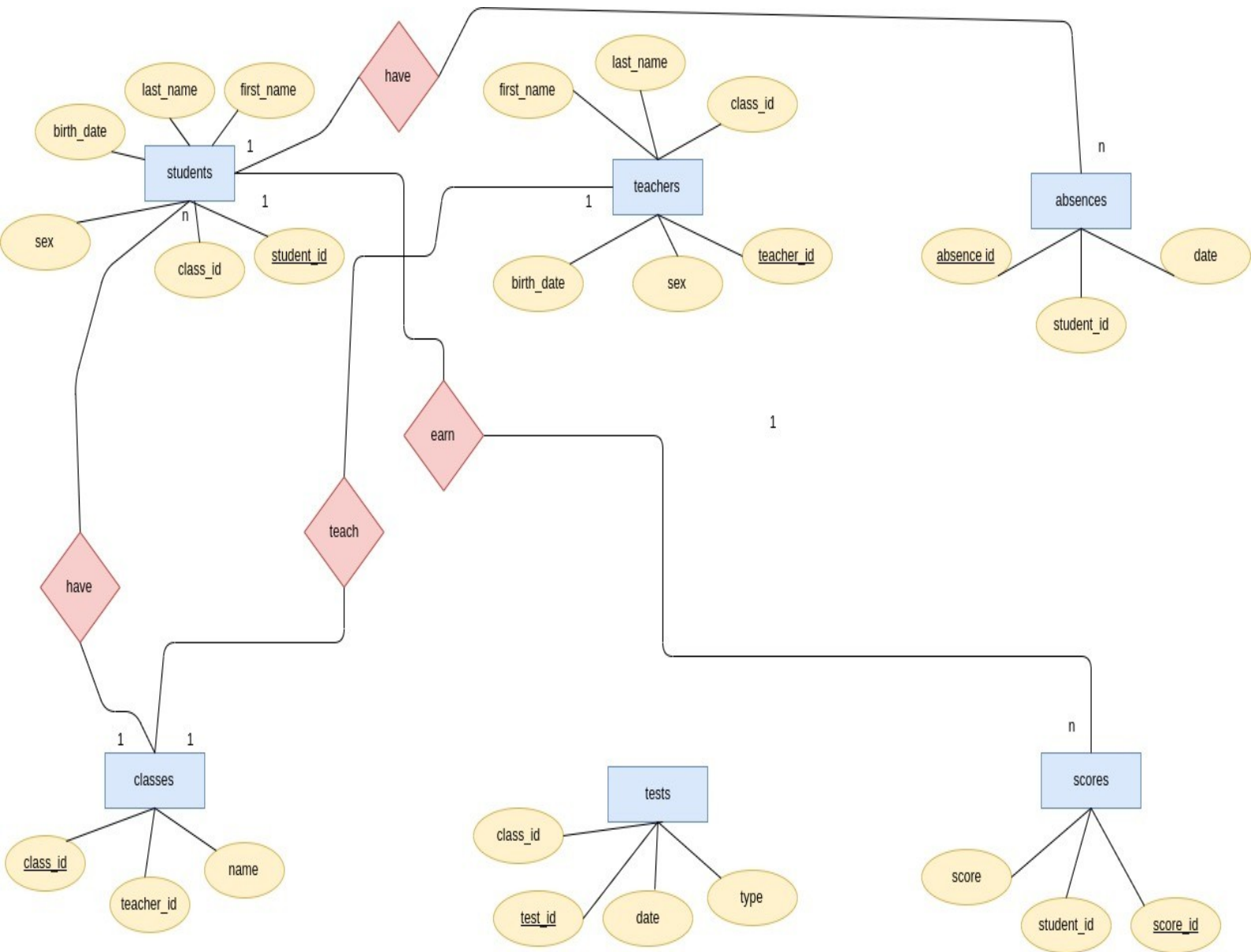
mysql> 
```

I also wrote the same trigger to delete all corresponding scores of that student if the student was deleted from the students table.

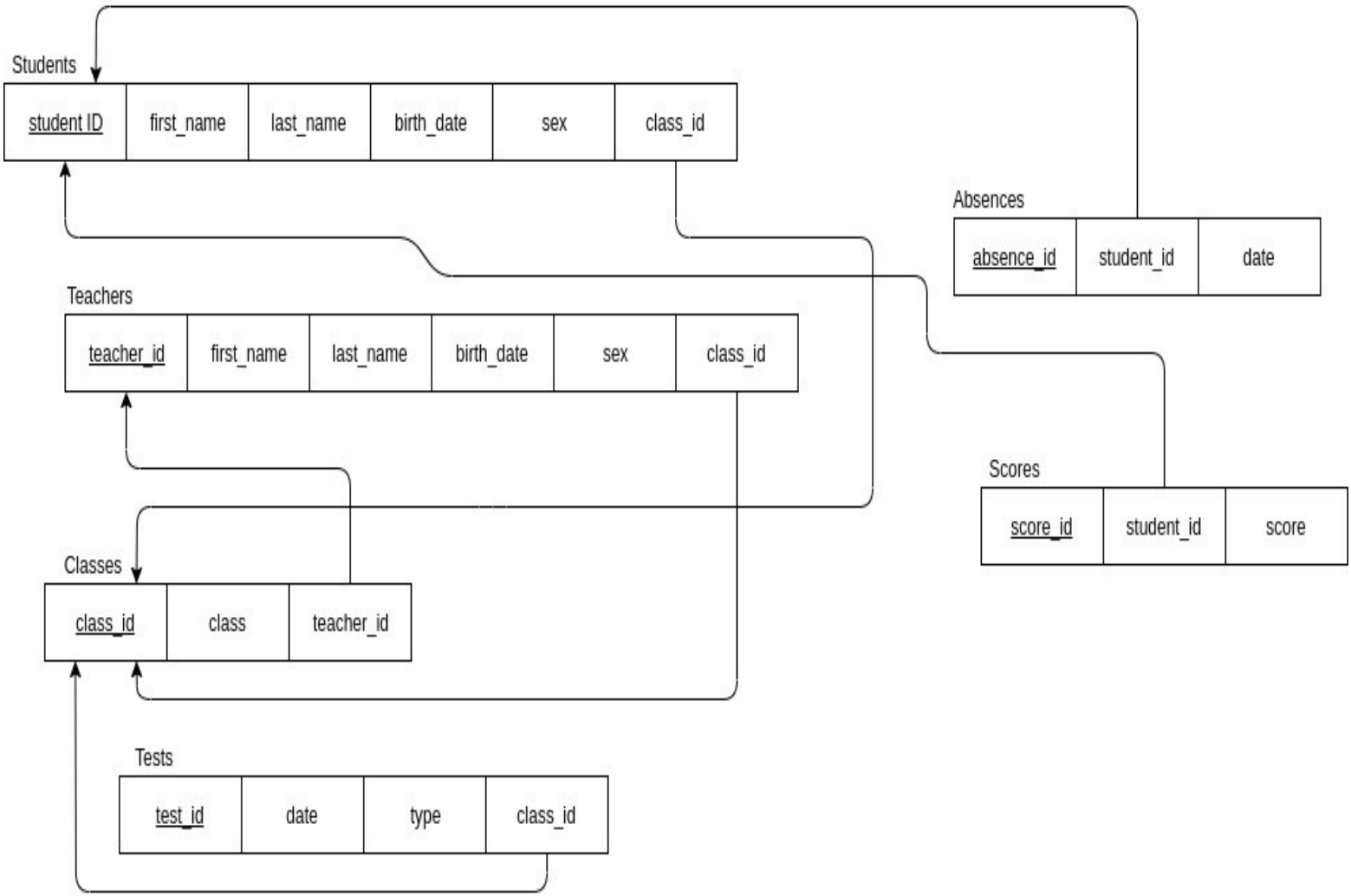
```
DELIMITER //
CREATE TRIGGER updateStudentScores
AFTER DELETE ON students
FOR EACH ROW
BEGIN
DELETE FROM scores
WHERE scores.student_id = old.student_id;
END//
DELIMITER ;
```

5 – Diagrams

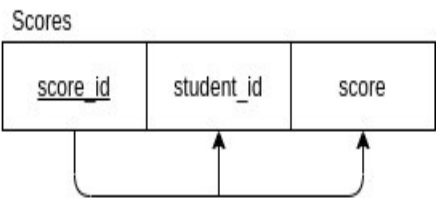
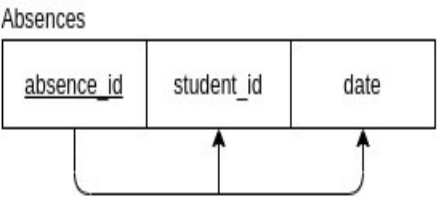
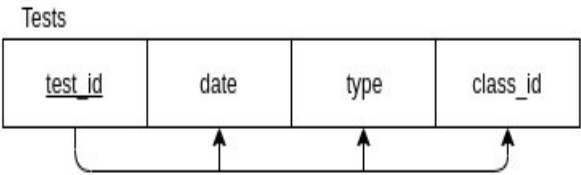
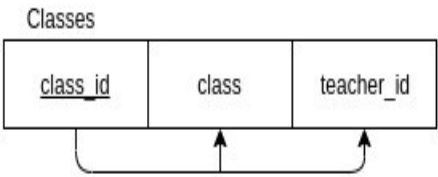
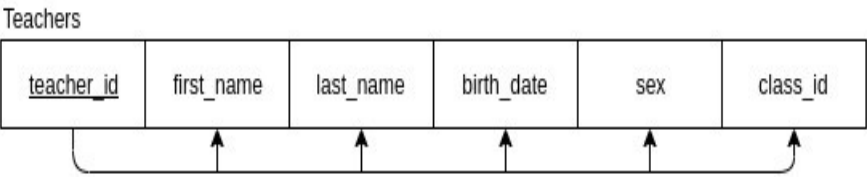
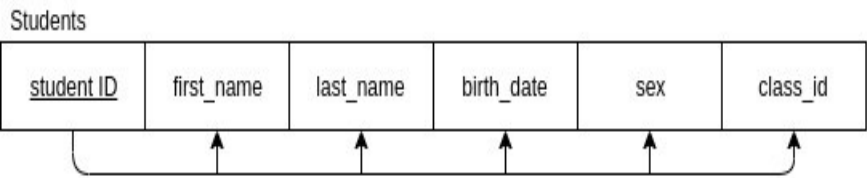
Entity Relationship Diagram



Relational Schema



Functional Dependency



6 – Appendix

Here is the code that I used to implement my database. It is important to note that when implementing this database on mysql using the command line on my terminal I found it most efficient to create tables, insert the data and then add foreign keys to the tables as to avoid foreign key constraints when inserting values into the tables.

```
CREATE TABLE students(  
first_name VARCHAR(40) NOT NULL,  
last_name VARCHAR(40) NOT NULL,  
birth_date VARCHAR(20) NOT NULL,  
sex VARCHAR(10) NOT NULL,  
class_id INTEGER NOT NULL,  
student_id INTEGER NOT NULL,  
Primary Key(student_id)  
);
```

```
CREATE TABLE teachers(  
first_name VARCHAR(40) NOT NULL,  
last_name VARCHAR(40) NOT NULL,  
birth_date VARCHAR(20) NOT NULL,  
sex VARCHAR(10) NOT NULL,  
class_id INTEGER NOT NULL,  
teacher_id INTEGER NOT NULL,  
Primary Key(teacher_id)  
);
```

```
CREATE TABLE classes(  
name VARCHAR(30) NOT NULL,  
teacher_id INTEGER NOT NULL,  
class_id INTEGER NOT NULL,  
Primary Key(class_id)  
);
```

```
CREATE TABLE tests(  
date VARCHAR(30) NOT NULL,  
class_id INTEGER NOT NULL,  
test_id INTEGER NOT NULL,  
Primary Key(test_id)  
);
```

```
CREATE TABLE absences(  
  student_id INTEGER NOT NULL,  
  date VARCHAR(30) NOT NULL,  
  absence_id INTEGER NOT NULL,  
  Primary Key(absence_id)  
);
```

```
CREATE TABLE scores(  
  student_id INTEGER NOT NULL,  
  score INTEGER NOT NULL,  
  test_id INTEGER NOT NULL,  
  score_id INTEGER NOT NULL,  
  Primary Key(score_id)  
);
```

```
CREATE TRIGGER upperName  
BEFORE INSERT ON students  
FOR EACH ROW  
SET NEW.first_name = UPPER(NEW.first_name);
```

```
CREATE TRIGGER upperLastName  
BEFORE INSERT ON students  
FOR EACH ROW  
SET NEW.last_name = UPPER(NEW.last_name);
```

```
DELIMITER //  
CREATE TRIGGER updateStudentAbsences  
AFTER DELETE ON students  
FOR EACH ROW  
BEGIN  
  DELETE FROM absences  
  WHERE absences.student_id = old.student_id;  
END//  
DELIMITER ;
```

```
DELIMITER //  
CREATE TRIGGER updateStudentScores  
AFTER DELETE ON students  
FOR EACH ROW  
BEGIN  
  DELETE FROM scores  
  WHERE scores.student_id = old.student_id;  
END//  
DELIMITER ;
```

```
CREATE VIEW allStudents AS
```

```
SELECT Concat(first_name, ',',last_name, ',',birth_date) AS class FROM students;
```

```
CREATE VIEW class5 AS
```

```
SELECT Concat(first_name, ',',last_name, ',',birth_date) AS class FROM students WHERE  
class_id = 5;
```

```
INSERT INTO students VALUES('John', 'Black', '27/08/2016', 'male', 1, 1);  
INSERT INTO students VALUES('James', 'Brown', '22/04/2016', 'male', 1, 2);  
INSERT INTO students VALUES('Aaron', 'Green', '16/05/2015', 'male', 2, 3);  
INSERT INTO students VALUES('Jen', 'White', '29/12/2015', 'female', 2, 4);  
INSERT INTO students VALUES('Alice', 'Reynolds', '04/07/2014', 'female', 3, 5);  
INSERT INTO students VALUES('Peter', 'Bates', '23/05/2014', 'male', 3, 6);  
INSERT INTO students VALUES('Anne Ryan', '24/10/2013', 'female', 4, 7);  
INSERT INTO students VALUES('Paddy', 'Byrne', '15/05/2013', 'male', 4, 8);  
INSERT INTO students VALUES('Regina', 'Harford', '17/07/2012', 'female', 5, 9);  
INSERT INTO students VALUES('Jimmy', 'Wilkinson', '27/08/2012', 'male', 5, 10);  
INSERT INTO students VALUES('Ciara', 'Hyland', '29/09/2011', 'female', 6, 11);  
INSERT INTO students VALUES('Luke', 'Doyle', '12/12/2011', 'male', 6, 12);  
INSERT INTO students VALUES('Diana', 'Williams', '14/01/2010', 'female', 7, 13);  
INSERT INTO students VALUES('Wesley', 'Potter', '19/03/2010', 'male', 7, 14);  
INSERT INTO students VALUES('Stefanie', 'Kavanagh', '20/12/2009', 'female', 8, 15);  
INSERT INTO students VALUES('David', 'Omoregie', '12/04/2009', 'male', 8, 16);
```

```
INSERT INTO teachers VALUES('Luke', 'Murphy', '01/03/1967', 'male', 1, 1);  
INSERT INTO teachers VALUES('Lucy', 'Monroe', '08/11/1990', 'female', 2, 2);  
INSERT INTO teachers VALUES('Leon', 'Dunne', '09/03/1983', 'male', 3, 3);  
INSERT INTO teachers VALUES('Laura', 'Kelly', '21/03/1960', 'female', 4, 4);  
INSERT INTO teachers VALUES('Shay', 'Ennis', '17/03/1991', 'male', 5, 5);  
INSERT INTO teachers VALUES('Neil', 'Maddy', '15/03/1988', 'male', 6, 6);  
INSERT INTO teachers VALUES('Lauren', 'Dunphy', '30/03/1996', 'female', 7, 7);  
INSERT INTO teachers VALUES('Eric', 'Mooney', '04/03/1975', 'male', 8, 8);
```

```
INSERT INTO classes VALUES('Junior Infants', 1, 1);  
INSERT INTO classes VALUES('Senior Infants', 2, 2);  
INSERT INTO classes VALUES('1st Class', 3, 3);  
INSERT INTO classes VALUES('2nd Class', 4, 4);  
INSERT INTO classes VALUES('3rd Class', 5, 5);  
INSERT INTO classes VALUES('4th Class', 6, 6);  
INSERT INTO classes VALUES('5th Class', 7, 7);  
INSERT INTO classes VALUES('6th Class', 8, 8);
```

```
INSERT INTO tests VALUES('11/12/2020', 1, 5);
```

```
INSERT INTO tests VALUES('26/11/2020', 1, 4);
INSERT INTO tests VALUES('21/10/2020', 1, 3);
INSERT INTO tests VALUES('24/09/2020', 1, 2);
INSERT INTO tests VALUES('29/08/2020', 1, 1);
```

```
INSERT INTO absences VALUES(1, '15/09/2020', 1);
INSERT INTO absences VALUES(2, '22/11/2020', 2);
INSERT INTO absences VALUES(3, '07/12/2020', 3);
INSERT INTO absences VALUES(4, '30/09/2020', 4);
INSERT INTO absences VALUES(5, '28/09/2020', 5);
```

```
INSERT INTO scores VALUES(1, 77, 1, 1);
INSERT INTO scores VALUES(2, 87, 1, 2);
INSERT INTO scores VALUES(3, 79, 1, 3);
INSERT INTO scores VALUES(4, 97, 1, 4);
INSERT INTO scores VALUES(5, 90, 1, 5);
```

```
ALTER TABLE students ADD Foreign Key(class_id) References classes(class_id);
ALTER TABLE teachers ADD Foreign Key(class_id) References classes(class_id);
ALTER TABLE classes ADD Foreign Key(teacher_id) References teachers(teacher_id);
ALTER TABLE tests ADD Foreign Key(class_id) References classes(class_id);
ALTER TABLE absences ADD Foreign Key(student_id) References students(student_id);
ALTER TABLE scores ADD Foreign Key(student_id) References students(student_id);
```