CSC 675/775 : Introduction to Database Systems

Task 1 (Phase 1)

Student Management System

Name of group members:

Ramit Singh - Section 1

Banaz Sinjary - Section 1

Pragati Makani - Section 2

Aaron Singh - Section 1

Adam Garcia - Section 1

Sanjana Devi - Section 1

Project Description

Student Teacher interaction application. We will be making an application that allows students to view their classes with their corresponding grades, and allows teachers who teach the class to update the grade of each student. Our database will have 4 entities, Student, Teacher, Class, and Grade which will store the following data:

Entities	Data (underline is primary key)
Student	student_id, name, data_of_birth, grade_level, phone_number, email, address
Teacher	teacher_id, name, phone_number, email, address, classes_taught
Classes	class_id, name, description, teacher_id, students_enrolled
Grades	grade_id, student_id, class_id, teacher_id, grade

They will be related by the following relationships:

Relationships	Entities Involved and How
Teaches	A **Teacher teaches** a **Class** and **teaches Students**
Gives	A **Teacher gives** a **Grade**
Gets	A **Student gets** a **Grade**

Database Description

1. Students table: This table will include fields for student_id, name, date_of_birth, grade_level, phone_number, email, and address.
2. Teachers table: This table will include fields for teacher_id, name, phone_number, email, address, and classes they teach.
3. Classes table: This table will include fields for class_id, name, description, teacher_id, and students enrolled in the class.
4. Grades table: This table will include fields for grade_id, student_id, class_id, teacher_id, and grade_score.

Data requirements

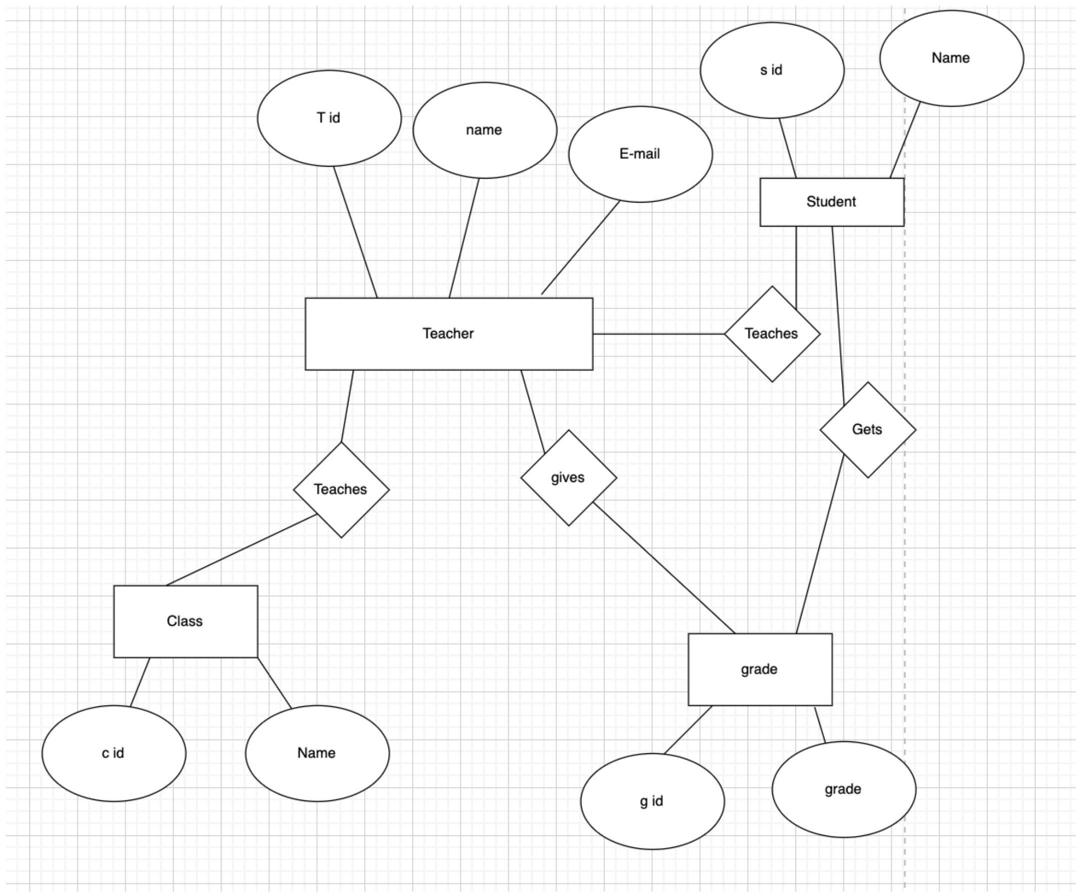
The data requirements for each table are as follows:

1. Students table: We need to store the data for all students enrolled in the school, including their name, date of birth, grade level, and contact information.
2. Teachers table: We need to store the data for all teachers employed by the school, including their name, contact information, and the classes they teach.
3. Classes table: We need to store the data for all classes offered by the school, including the class name, description, teacher, and students enrolled in each class.
4. Grades table: We need to store the data for all grades assigned to students in each class, including the student, class, teacher, and grade score.

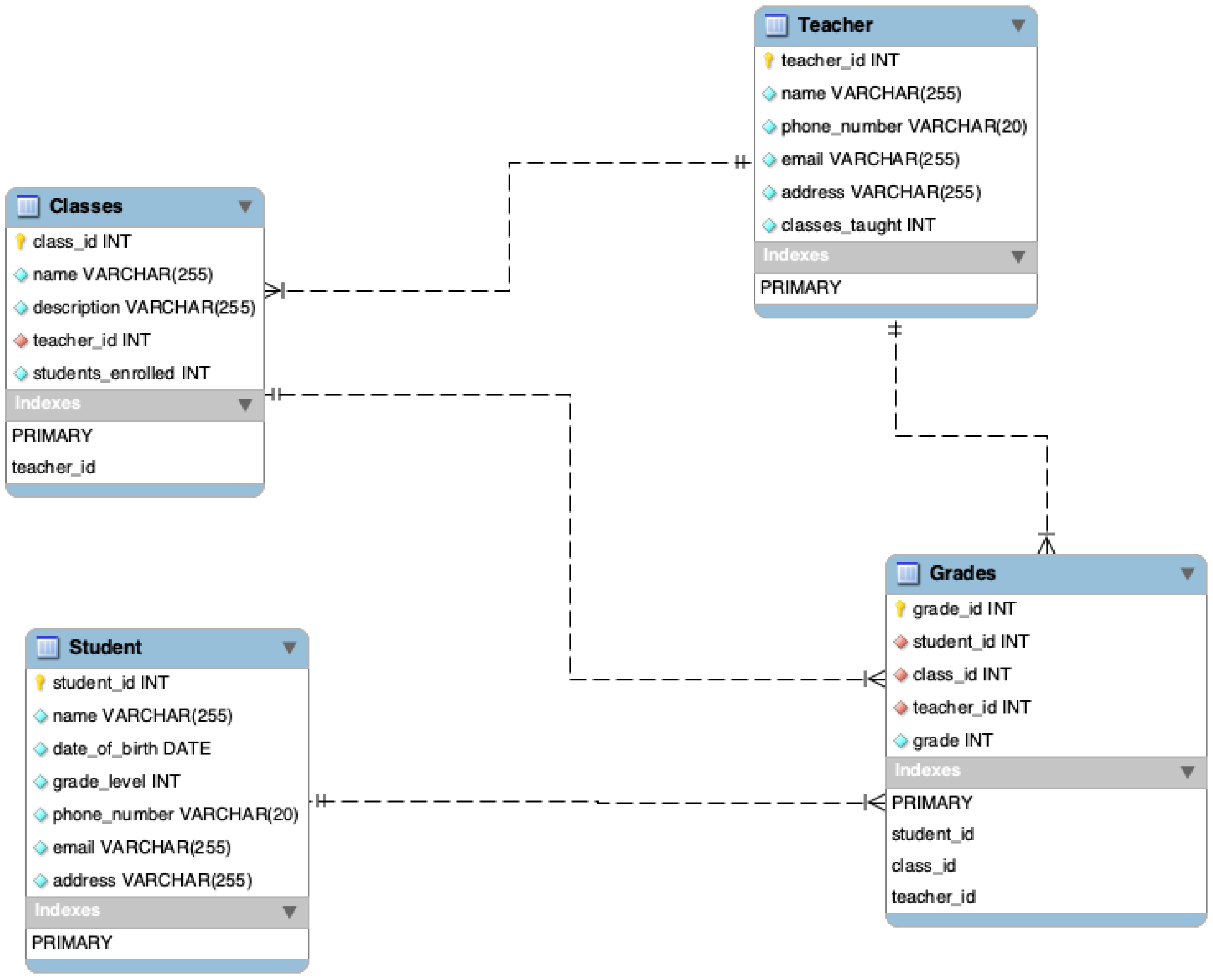
Constraints on the data

1. Each student must have a unique identifier and name.
2. Each teacher must have a unique identifier and name.
3. Each class must have a unique identifier and name.
4. Each grade must have a unique identifier and be associated with a specific student, class, and teacher.

ER Diagram


s id
Name
T id
name
E-mail
Student
Teacher
Teaches
Gets
Teaches
gives
Class
grade
c id
Name
g id
grade

Relational Schemas



CSC 675/775 : Introduction to Database Systems

Task 2 (Phase 2)

Student Management System

Name of group members:

Ramit Singh - Section 1

Banaz Sinjary - Section 1

Pragati Makani - Section 2

Aaron Singh - Section 1

Adam Garcia - Section 1

Sanjana Devi - Section 1

Tables, indexes & view statements

Create Tables

- **Students**:

```
CREATE TABLE Students (
 -- `student_id` is the primary key of the table.
 -- This column uniquely identifies each student.
 student_id INT PRIMARY KEY,
 -- `name` is the student's name.
 -- This column cannot be null.
 name VARCHAR(50) NOT NULL,
 -- `date_of_birth` is the student's date of birth.
 date_of_birth DATE,
 -- `grade_level` is the student's grade level.
 grade_level INT,
 -- `phone_number` is the student's phone number.
 phone_number VARCHAR(20),
 -- `email` is the student's email address.
 email VARCHAR(50),
 -- `address` is the student's address.
 address VARCHAR(200)
);
```

- **Teachers**:

```
CREATE TABLE Teachers (
    -- `teacher_id` is the primary key of the table.
     -- This column uniquely identifies each teacher.
    teacher_id INT PRIMARY KEY,
    -- `name` is the teacher's name.
    -- This column cannot be null.
    name VARCHAR(50) NOT NULL,
    -- `phone_number` is the teacher's phone number.
    phone_number VARCHAR(20),
    -- `email` is the teacher's email address.
     email VARCHAR(50),
     -- `address` is the teacher's address.
     address VARCHAR(200),
     -- `classes_taught` is a list of the classes that the teacher
teaches.
     classes_taught VARCHAR(200)
);
```

- **Classes**:

```
CREATE TABLE Classes (
       -- `class_id` is the primary key of the table.
 -- This column uniquely identifies each class.
      class_id INT PRIMARY KEY,
      -- `name` is the name of the class.
      -- This column cannot be null.
      name VARCHAR(50) NOT NULL,
       -- `description` is a description of the class.
       description VARCHAR(200),
 -- `teacher_id` is the ID of the teacher who teaches the class.
       teacher_id INT,
       -- This foreign key constraint ensures that the `teacher_id` column
references a valid row in the `Teachers` table.
      FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
CREATE TABLE Classes (
 class_id INT PRIMARY KEY,
 name VARCHAR(50) NOT NULL,
 description VARCHAR(200),
 teacher_id INT,
 FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
```

- **Grades**:

```
CREATE TABLE Grades (
       -- `grade_id` is the primary key of the table.
       -- This column uniquely identifies each grade.
       grade_id INT PRIMARY KEY,
 -- `student_id` is the ID of the student who received the grade.
 student_id INT,
       -- `class_id` is the ID of the class that the student took.
       class_id INT,
       -- `teacher_id` is the ID of the teacher who taught the class.
 teacher_id INT,
       -- `grade_score` is the student's grade in the class.
       grade_score INT,
      -- This foreign key constraint ensures that the `student_id` column
references a valid row in the `Students` table.
       FOREIGN KEY (student_id) REFERENCES Students(student_id),
      -- This foreign key constraint ensures that the `class_id` column
references a valid row in the `Classes` table.
```

```
      FOREIGN KEY (class_id) REFERENCES Classes(class_id),
 -- This foreign key constraint ensures that the `teacher_id` column references
a valid row in the `Teachers` table.
      FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
CREATE TABLE Grades (
 grade_id INT PRIMARY KEY,
 student_id INT,
 class_id INT,
 teacher_id INT,
 grade_score INT,
 FOREIGN KEY (student_id) REFERENCES Students(student_id),
 FOREIGN KEY (class_id) REFERENCES Classes(class_id),
 FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
```

- **Create Index**:

```
CREATE INDEX grade_score_idx ON Grades (grade_score);
CREATE INDEX grade_score_idx ON Grades (grade_score);
```

- **Create view statements**:

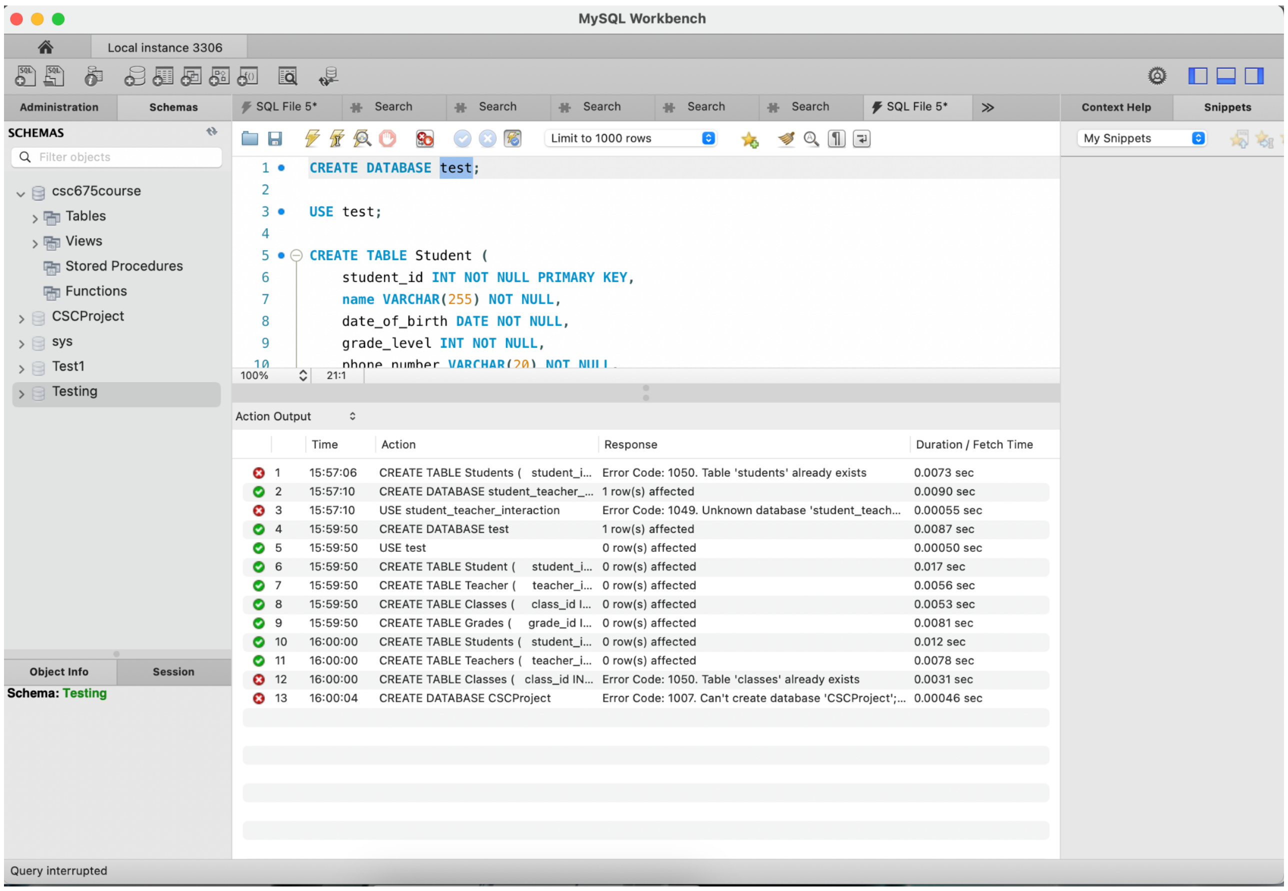
```
CREATE VIEW high_grades AS
 SELECT s.name AS student_name, g.grade_score
 FROM Students s
 JOIN Grades g
CREATE VIEW high_grades AS
 SELECT s.name AS student_name, g.grade_score
 FROM Students s
 JOIN Grades g
 ON s.student_id = g.student_id
 WHERE g.grade_score > 90;
```

- **Adding Teachers**

```
      USE Testing;
```

```

INSERT INTO teachers (name, phone_number, email, address)
VALUES
 ('Anna Davis', '1980-05-12', 'English', '(123) 456-7890',
'anna.davis@email.com', '123 Main Street, Anytown, CA 91234'),
 ('Benjamin Green', '1985-08-21', 'Math', '(555) 678-9012',
'benjamin.green@email.com', '456 Elm Street, Anytown, CA 91234'),
 ('Carla Hernandez', '1978-02-14', 'Spanish', '(111) 222-3333',
'carla.hernandez@email.com', '789 Oak Street, Anytown, CA 91234'),
 ('David Lee', '1982-07-06', 'Science', '(555) 555-5555',
'david.lee@email.com', '234 Maple Street, Anytown, CA 91234'),
 ('Emily Brown', '1979-11-18', 'Social Studies', '(222) 333-4444',
'emily.brown@email.com', '345 Pine Street, Anytown, CA 91234'),
 ('Frank Johnson', '1981-04-30', 'Physical Education', '(333) 444-5555',
'frank.johnson@email.com', '456 Cedar Street, Anytown, CA 91234'),
 ('Gina Rodriguez', '1976-09-01', 'Art', '(444) 555-6666',
'gina.rodriguez@email.com', '567 Oak Street, Anytown, CA 91234'),
 ('Harry Lee', '1984-01-25', 'Math', '(555) 666-7777',
'harry.lee@email.com', '678 Elm Street, Anytown, CA 91234'),
 ('Ivy Chen', '1977-06-07', 'Science', '(666) 777-8888',
'ivy.chen@email.com', '789 Pine Street, Anytown, CA 91234'),
 ('Jake Williams', '1980-10-19', 'Social Studies', '(777) 888-9999',
'jake.williams@email.com', '890 Cedar Street, Anytown, CA 91234'),
 ('Karen Taylor', '1979-03-03', 'English', '(888) 999-0000',
'karen.taylor@email.com', '901 Oak Street, Anytown, CA 91234'),
 ('Liam Brown', '1976-07-16', 'Physical Education', '(111) 222-3333',
'liam.brown@email.com', '123 Elm Street, Anytown, CA 91234'),
 ('Mary Johnson', '1983-12-28', 'Spanish', '(222) 333-4444',
'mary.johnson@email.com', '234 Pine Street, Anytown, CA 91234'),
 ('Nick Perez', '1978-05-12', 'Art', '(333) 444-5555',
'nick.perez@email.com', '345 Cedar Street, Anytown, CA 91234'),
 ('Olivia Davis', '1981-10-25', 'Science', '(444) 555-6666',
'olivia.davis@email.com', '456 Oak Street, Anytown, CA 91234');
```

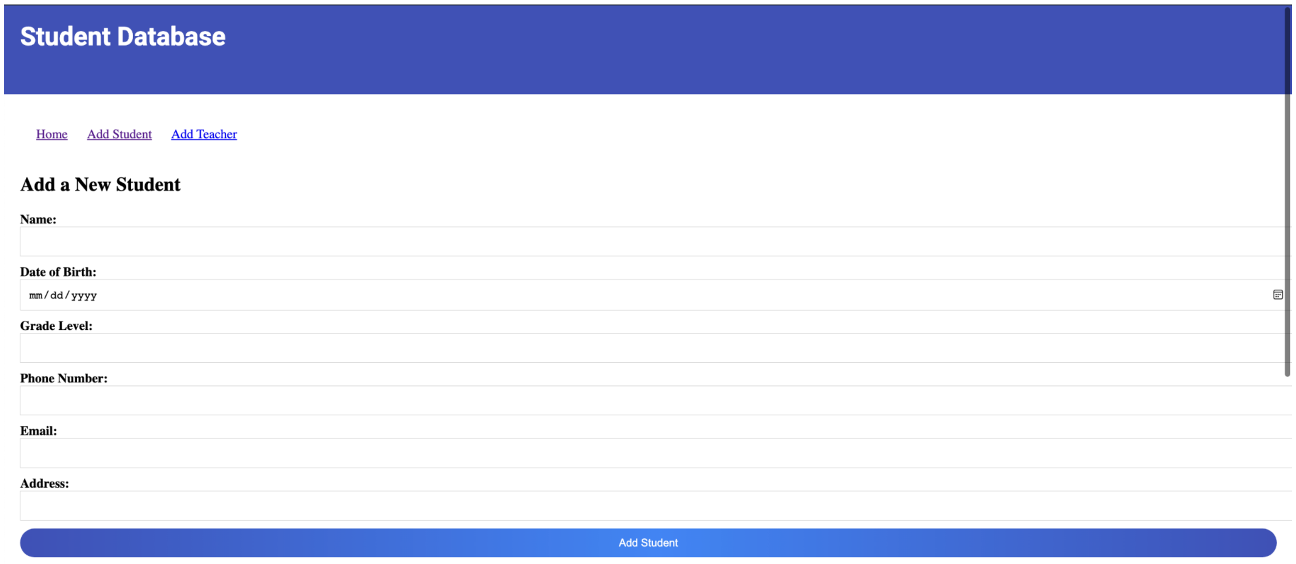
Screenshot of the SQL results:



```sql
CREATE DATABASE test;

USE test;

CREATE TABLE Student (
    student_id INT NOT NULL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    date_of_birth DATE NOT NULL,
    grade_level INT NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
```

		Time	Action	Response	Duration / Fetch Time
✗	1	15:57:06	CREATE TABLE Students (student_i...	Error Code: 1050. Table 'students' already exists	0.0073 sec
✓	2	15:57:10	CREATE DATABASE student_teacher_...	1 row(s) affected	0.0090 sec
✗	3	15:57:10	USE student_teacher_interaction	Error Code: 1049. Unknown database 'student_teach...	0.00055 sec
✓	4	15:59:50	CREATE DATABASE test	1 row(s) affected	0.0087 sec
✓	5	15:59:50	USE test	0 row(s) affected	0.00050 sec
✓	6	15:59:50	CREATE TABLE Student (student_i...	0 row(s) affected	0.017 sec
✓	7	15:59:50	CREATE TABLE Teacher (teacher_i...	0 row(s) affected	0.0056 sec
✓	8	15:59:50	CREATE TABLE Classes (class_id I...	0 row(s) affected	0.0053 sec
✓	9	15:59:50	CREATE TABLE Grades (grade_id I...	0 row(s) affected	0.0081 sec
✓	10	16:00:00	CREATE TABLE Students (student_i...	0 row(s) affected	0.012 sec
✓	11	16:00:00	CREATE TABLE Teachers (teacher_i...	0 row(s) affected	0.0078 sec
✗	12	16:00:00	CREATE TABLE Classes (class_id IN...	Error Code: 1050. Table 'classes' already exists	0.0031 sec
✗	13	16:00:04	CREATE DATABASE CSCProject	Error Code: 1007. Can't create database 'CSCProject';...	0.00046 sec

Web-app View front end to Home Page adding new Students



Add a New Student

Name:

Date of Birth: mm/dd/yyyy

Grade Level:

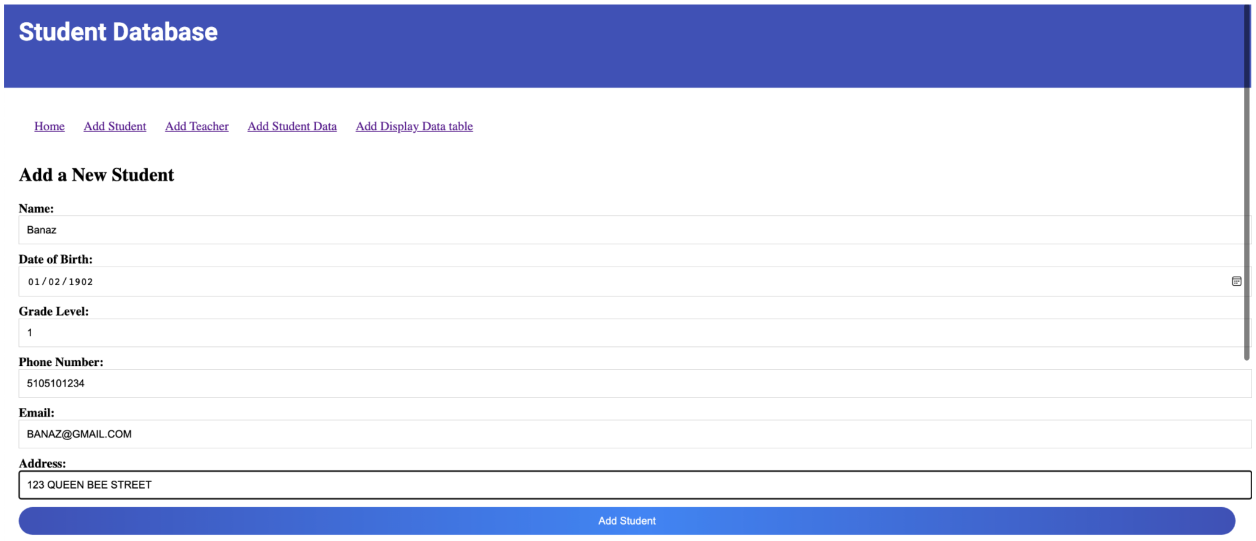
Phone Number:

Email:

Address:

Add Student

Web-app View front end to Home Page adding new Students


Student Database

Home Add Student Add Teacher Add Student Data Add Display Data table

Add a New Student

Name: Banaz

Date of Birth: 01/02/1902

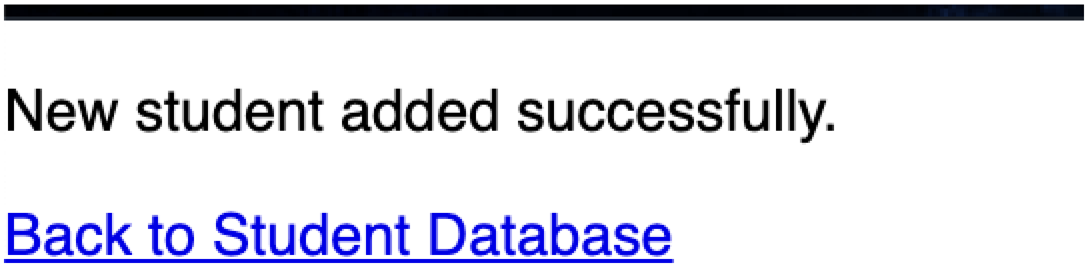
Grade Level: 1

Phone Number: 5105101234

Email: BANAZ@GMAIL.COM

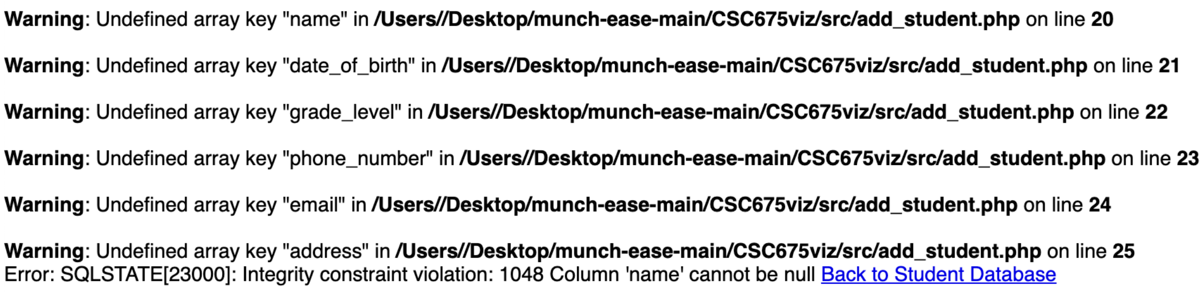
Address: 123 QUEEN BEE STREET

Web-app View front end to Home Page adding new Successfully added

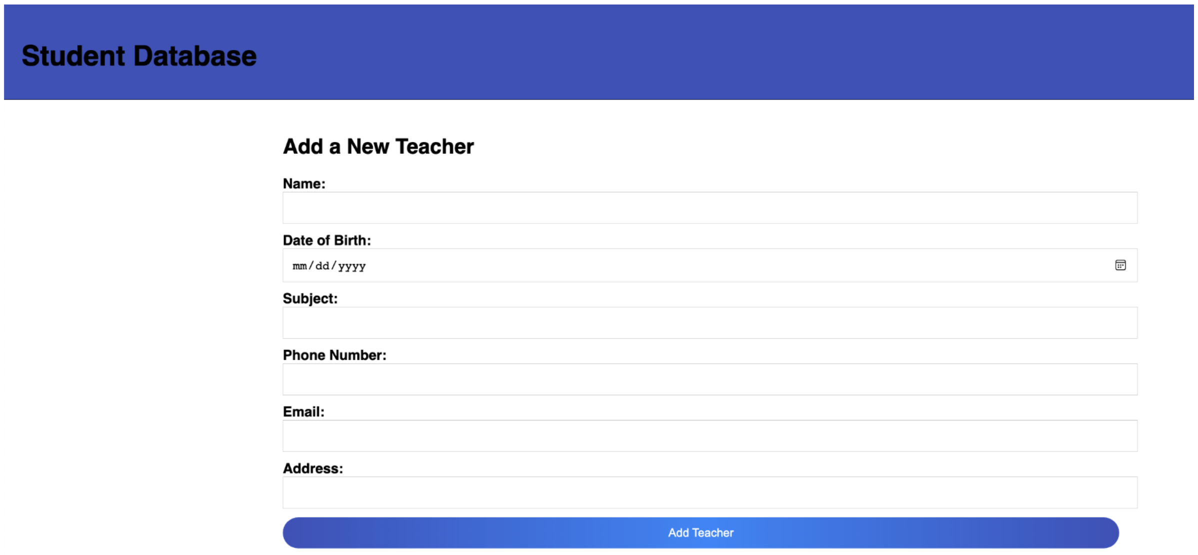

New student added successfully.

Back to Student Database

Web-app View front end to Home Page adding new failed


```
Warning: Undefined array key "name" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 20

Warning: Undefined array key "date_of_birth" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 21

Warning: Undefined array key "grade_level" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 22

Warning: Undefined array key "phone_number" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 23

Warning: Undefined array key "email" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 24

Warning: Undefined array key "address" in /Users//Desktop/munch-ease-main/CSC675viz/src/add_student.php on line 25
Error: SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'name' cannot be null Back to Student Database
```

<u>Web-app View front end to Teacher adding new Teachers</u>



Student Database

Add a New Teacher

Name:

Date of Birth: mm/dd/yyyy

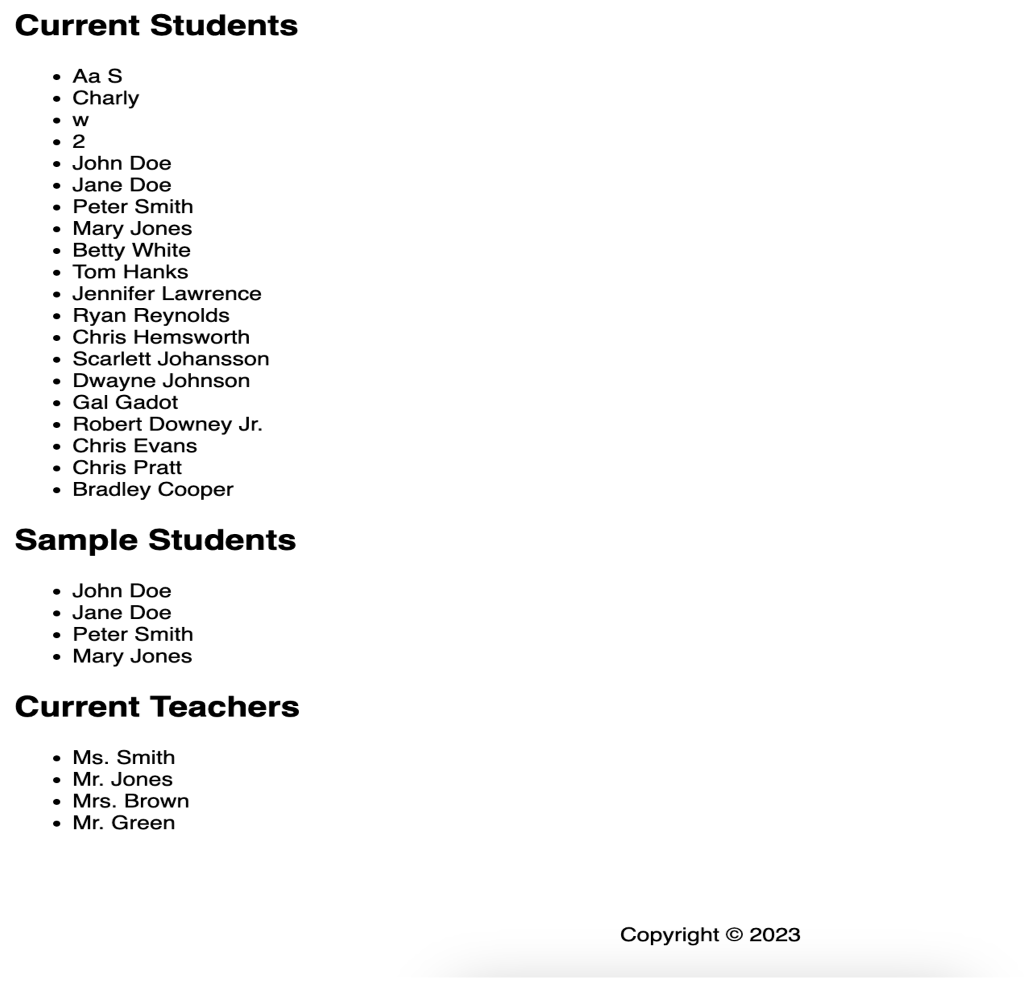
Subject:

Phone Number:

Email:

Address:

Web-app View front end to Teacher page with current students



Current Students

- Aa S
- Charly
- w
- 2
- John Doe
- Jane Doe
- Peter Smith
- Mary Jones
- Betty White
- Tom Hanks
- Jennifer Lawrence
- Ryan Reynolds
- Chris Hemsworth
- Scarlett Johansson
- Dwayne Johnson
- Gal Gadot
- Robert Downey Jr.
- Chris Evans
- Chris Pratt
- Bradley Cooper

Sample Students

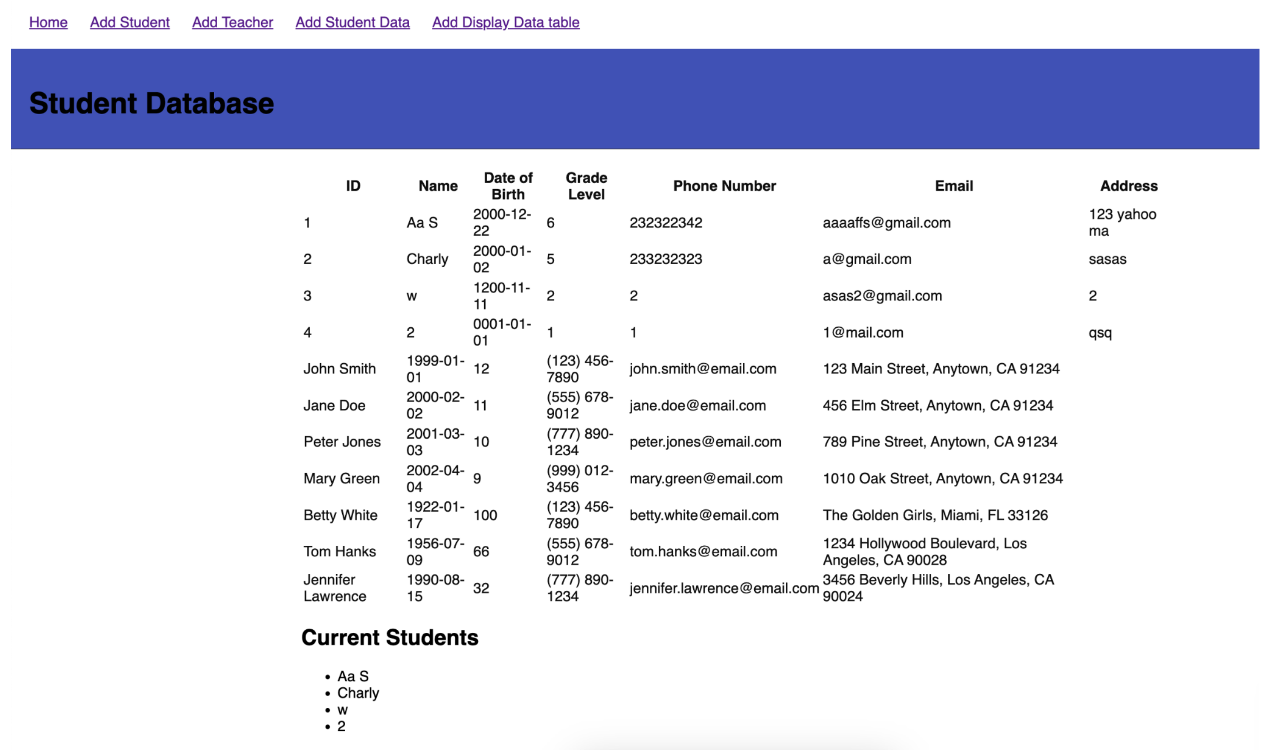
- John Doe
- Jane Doe
- Peter Smith
- Mary Jones

Current Teachers

- Ms. Smith
- Mr. Jones
- Mrs. Brown
- Mr. Green

Copyright © 2023

Web-app View front end to MySQL Query display in tables page with current students/ and PI data



Home   Add Student   Add Teacher   Add Student Data   Add Display Data table

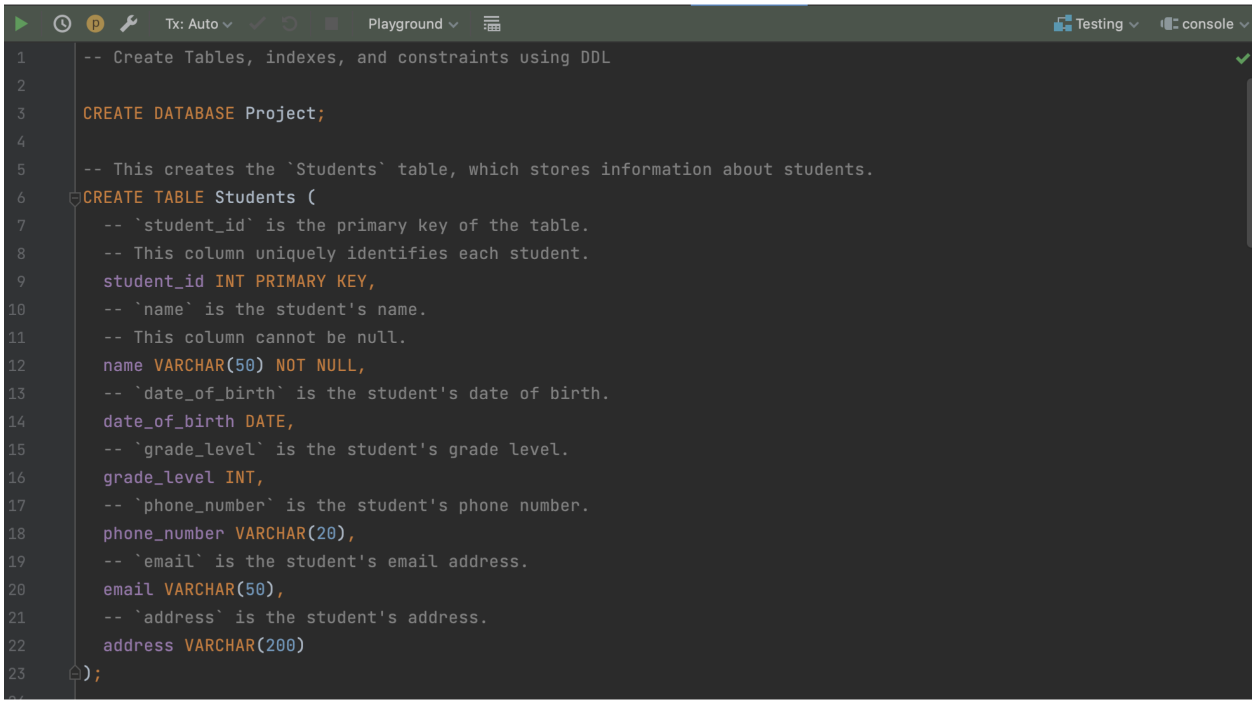
Student Database

ID	Name	Date of Birth	Grade Level	Phone Number	Email	Address
1	Aa S	2000-12-22	6	232322342	aaaaffs@gmail.com	123 yahoo ma
2	Charly	2000-01-02	5	233232323	a@gmail.com	sasas
3	w	1200-11-11	2	2	asas2@gmail.com	2
4	2	0001-01-01	1	1	1@mail.com	qsq
John Smith	1999-01-01	12	(123) 456-7890	john.smith@email.com	123 Main Street, Anytown, CA 91234	
Jane Doe	2000-02-02	11	(555) 678-9012	jane.doe@email.com	456 Elm Street, Anytown, CA 91234	
Peter Jones	2001-03-03	10	(777) 890-1234	peter.jones@email.com	789 Pine Street, Anytown, CA 91234	
Mary Green	2002-04-04	9	(999) 012-3456	mary.green@email.com	1010 Oak Street, Anytown, CA 91234	
Betty White	1922-01-17	100	(123) 456-7890	betty.white@email.com	The Golden Girls, Miami, FL 33126	
Tom Hanks	1956-07-09	66	(555) 678-9012	tom.hanks@email.com	1234 Hollywood Boulevard, Los Angeles, CA 90028	
Jennifer Lawrence	1990-08-15	32	(777) 890-1234	jennifer.lawrence@email.com	3456 Beverly Hills, Los Angeles, CA 90024	

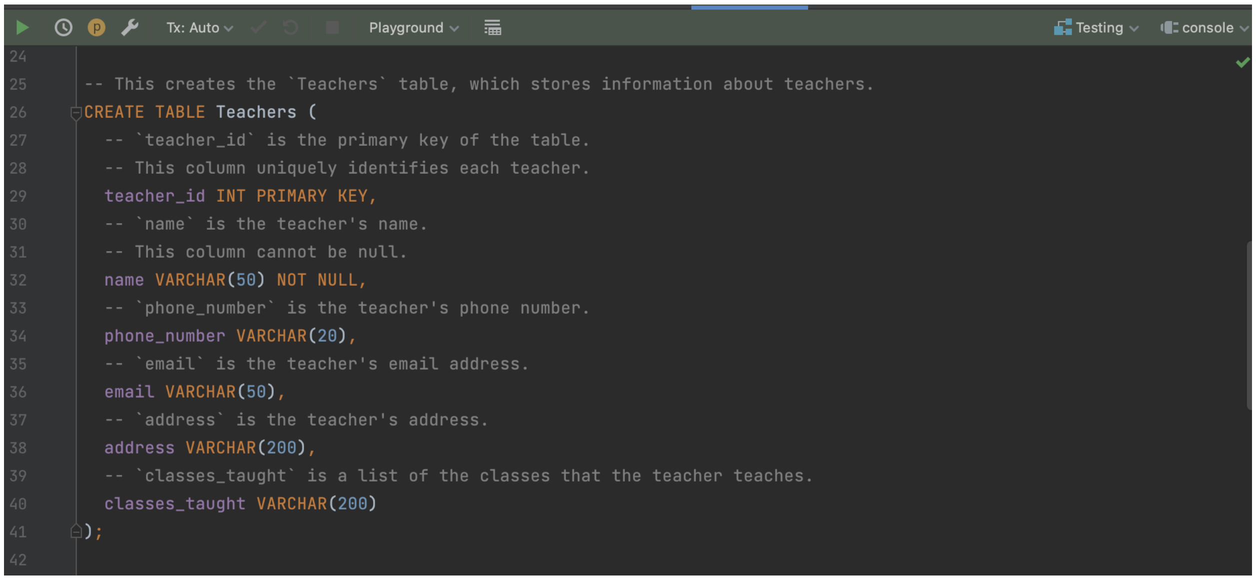
Current Students

- Aa S
- Charly
- w
- 2

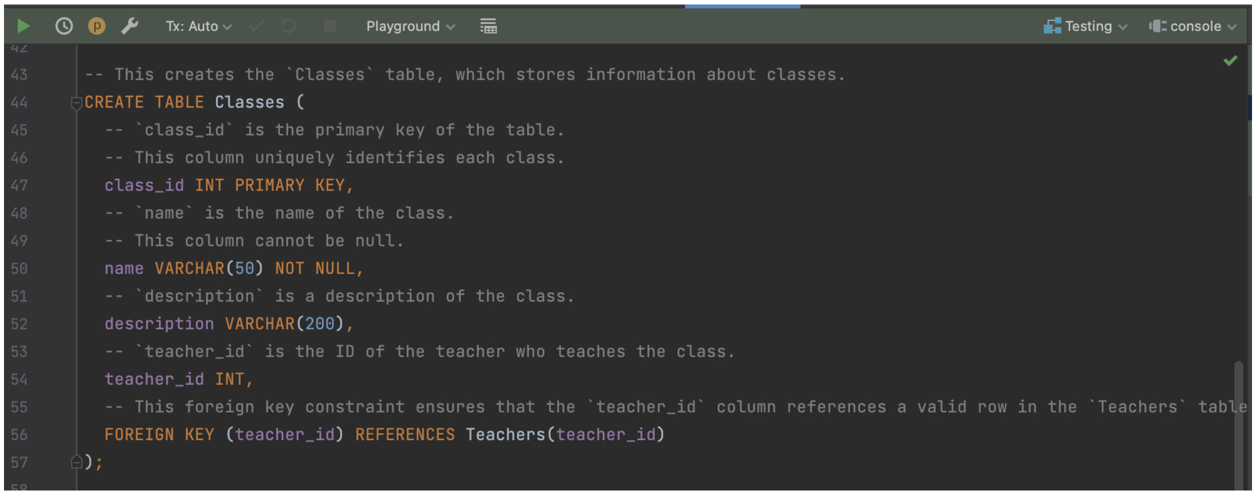
SQL Query ran for project



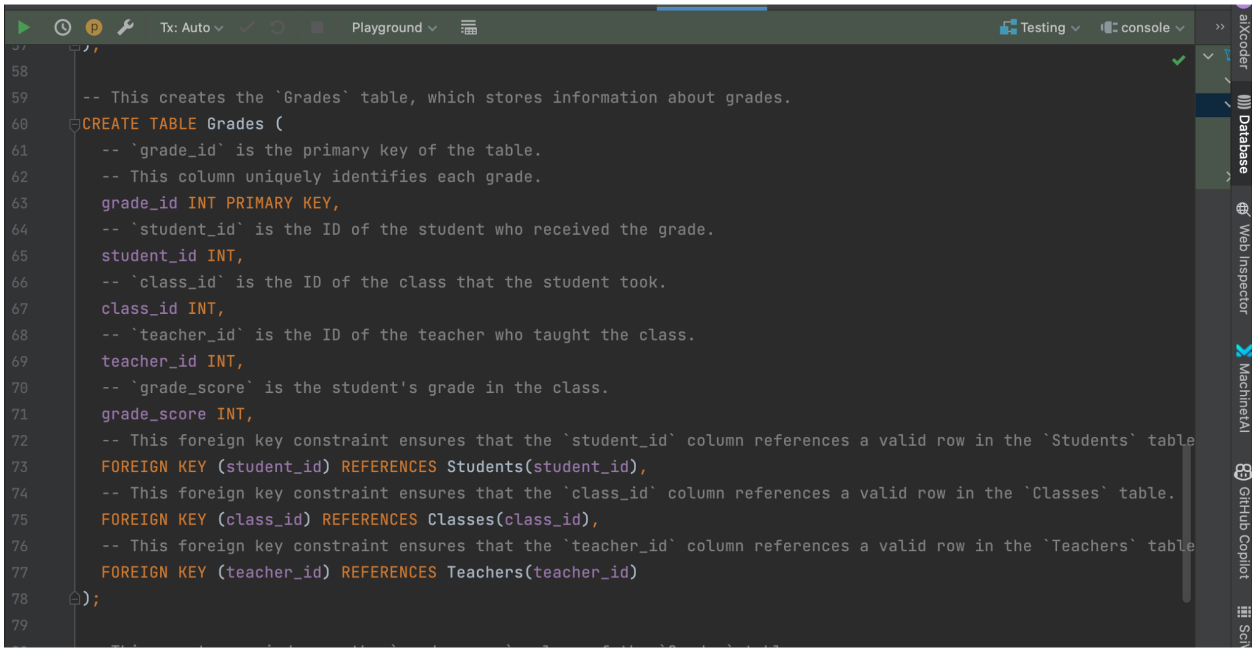
```sql
-- Create Tables, indexes, and constraints using DDL

CREATE DATABASE Project;

-- This creates the `Students` table, which stores information about students.
CREATE TABLE Students (
  -- `student_id` is the primary key of the table.
  -- This column uniquely identifies each student.
  student_id INT PRIMARY KEY,
  -- `name` is the student's name.
  -- This column cannot be null.
  name VARCHAR(50) NOT NULL,
  -- `date_of_birth` is the student's date of birth.
  date_of_birth DATE,
  -- `grade_level` is the student's grade level.
  grade_level INT,
  -- `phone_number` is the student's phone number.
  phone_number VARCHAR(20),
  -- `email` is the student's email address.
  email VARCHAR(50),
  -- `address` is the student's address.
  address VARCHAR(200)
);
```



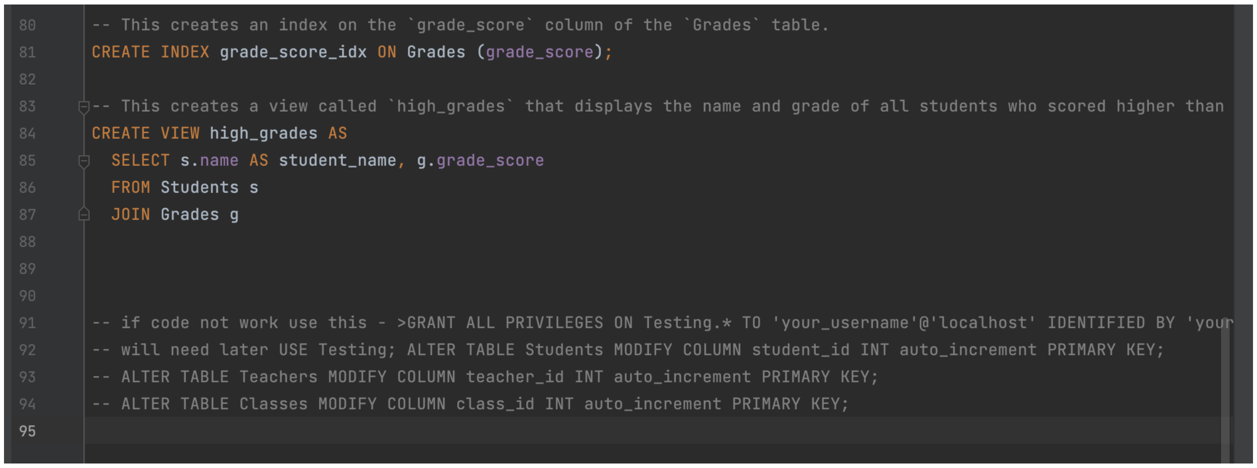
```sql

-- This creates the `Teachers` table, which stores information about teachers.
CREATE TABLE Teachers (
  -- `teacher_id` is the primary key of the table.
  -- This column uniquely identifies each teacher.
  teacher_id INT PRIMARY KEY,
  -- `name` is the teacher's name.
  -- This column cannot be null.
  name VARCHAR(50) NOT NULL,
  -- `phone_number` is the teacher's phone number.
  phone_number VARCHAR(20),
  -- `email` is the teacher's email address.
  email VARCHAR(50),
  -- `address` is the teacher's address.
  address VARCHAR(200),
  -- `classes_taught` is a list of the classes that the teacher teaches.
  classes_taught VARCHAR(200)
);

```



```sql
-- This creates the `Classes` table, which stores information about classes.
CREATE TABLE Classes (
  -- `class_id` is the primary key of the table.
  -- This column uniquely identifies each class.
  class_id INT PRIMARY KEY,
  -- `name` is the name of the class.
  -- This column cannot be null.
  name VARCHAR(50) NOT NULL,
  -- `description` is a description of the class.
  description VARCHAR(200),
  -- `teacher_id` is the ID of the teacher who teaches the class.
  teacher_id INT,
  -- This foreign key constraint ensures that the `teacher_id` column references a valid row in the `Teachers` table
  FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
```



```sql
-- This creates the `Grades` table, which stores information about grades.
CREATE TABLE Grades (
  -- `grade_id` is the primary key of the table.
  -- This column uniquely identifies each grade.
  grade_id INT PRIMARY KEY,
  -- `student_id` is the ID of the student who received the grade.
  student_id INT,
  -- `class_id` is the ID of the class that the student took.
  class_id INT,
  -- `teacher_id` is the ID of the teacher who taught the class.
  teacher_id INT,
  -- `grade_score` is the student's grade in the class.
  grade_score INT,
  -- This foreign key constraint ensures that the `student_id` column references a valid row in the `Students` table
  FOREIGN KEY (student_id) REFERENCES Students(student_id),
  -- This foreign key constraint ensures that the `class_id` column references a valid row in the `Classes` table.
  FOREIGN KEY (class_id) REFERENCES Classes(class_id),
  -- This foreign key constraint ensures that the `teacher_id` column references a valid row in the `Teachers` table
  FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id)
);
```



```sql
-- This creates an index on the `grade_score` column of the `Grades` table.
CREATE INDEX grade_score_idx ON Grades (grade_score);

-- This creates a view called `high_grades` that displays the name and grade of all students who scored higher than
CREATE VIEW high_grades AS
  SELECT s.name AS student_name, g.grade_score
  FROM Students s
  JOIN Grades g



-- if code not work use this - >GRANT ALL PRIVILEGES ON Testing.* TO 'your_username'@'localhost' IDENTIFIED BY 'your
-- will need later USE Testing; ALTER TABLE Students MODIFY COLUMN student_id INT auto_increment PRIMARY KEY;
-- ALTER TABLE Teachers MODIFY COLUMN teacher_id INT auto_increment PRIMARY KEY;
-- ALTER TABLE Classes MODIFY COLUMN class_id INT auto_increment PRIMARY KEY;
```