Creating a Students table with fields:

id (primary key, auto-increment)

name (varchar)

email (varchar)

age (int)

gender (varchar)

Here is the SQL statement to create the Students table:

```
CREATE TABLE Students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255),
    age INT,
    gender VARCHAR(10)
);
```

Setting up Spring Boot project and dependencies:

Create a new Spring Boot project using your preferred IDE or the Spring Boot CLI.

Add the following dependencies to your pom.xml file:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
```

```
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

These dependencies provide the necessary web and data access functionality, and use an H2 in-memory database for demonstration purposes.

Creating a Student entity class:

Create a new Java class called Student with the following fields, constructors, and getters/setters:

```
@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private int age;
    private String gender;
```

```
    public Student() {}

    public Student(String name, String email, int age, String gender) {
        this.name = name;
        this.email = email;
        this.age = age;
        this.gender = gender;
    }

    // getters and setters
}
```

Creating a Student repository interface:

Create a new Java interface called StudentRepository that extends the JpaRepository interface:

```
public interface StudentRepository extends JpaRepository<Student, Long> {}
```

Creating a RESTful web service controller:

Create a new Java class called StudentController with the following methods:

```
@RestController
public class StudentController {
    @Autowired
    private StudentRepository studentRepository;
```

```
    // GET all students
    @GetMapping("/students")
    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    // GET a specific student by ID
    @GetMapping("/students/{id}")
    public ResponseEntity<Student> getStudentById(@PathVariable Long id) {
        Optional<Student> student = studentRepository.findById(id);
        if (student.isPresent()) {
            return new ResponseEntity<>(student.get(), HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    // CREATE a new student
    @PostMapping("/students")
    public ResponseEntity<Student> createStudent(@RequestBody Student student) {
        Student newStudent = studentRepository.save(student);
        return new ResponseEntity<>(newStudent, HttpStatus.CREATED);
    }

    // UPDATE an existing student by ID
    @PutMapping("/students/{id}")
    public ResponseEntity<Student> updateStudent(@PathVariable Long id, @RequestBody Student
student) {
        Optional<Student> optionalStudent = studentRepository.findById(id);
```

```
if (optionalStudent.isPresent()) {
        Student updatedStudent = optionalStudent.get();
        updatedStudent.setName(student.getName());
        updatedStudent.setEmail
```