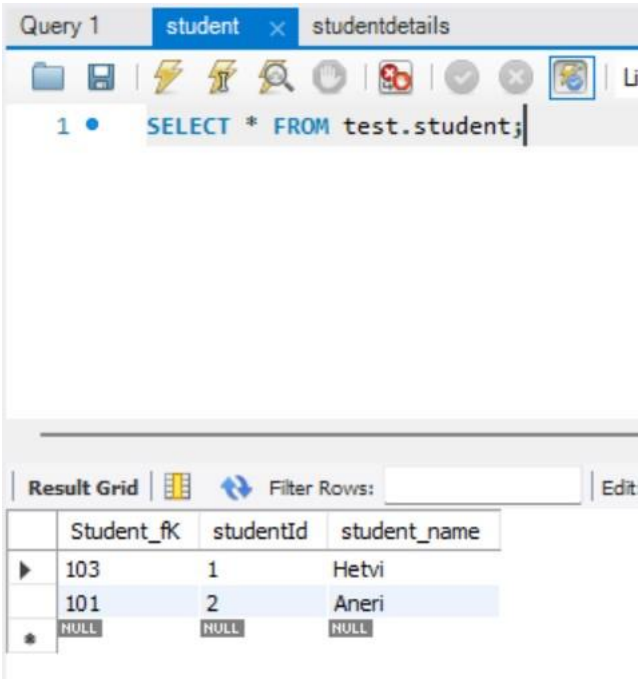
Unidirectional one-to-one mapping:

```java
@Entity
public class Student {
    @Id
    private long studentId;

    3 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "Student_fK")
    private StudentDetails studentDetails;

    no usages
    public long getStudentId() { return studentId; }

    2 usages
    public void setStudentId(long studentId) { this.studentId = studentId; }
```

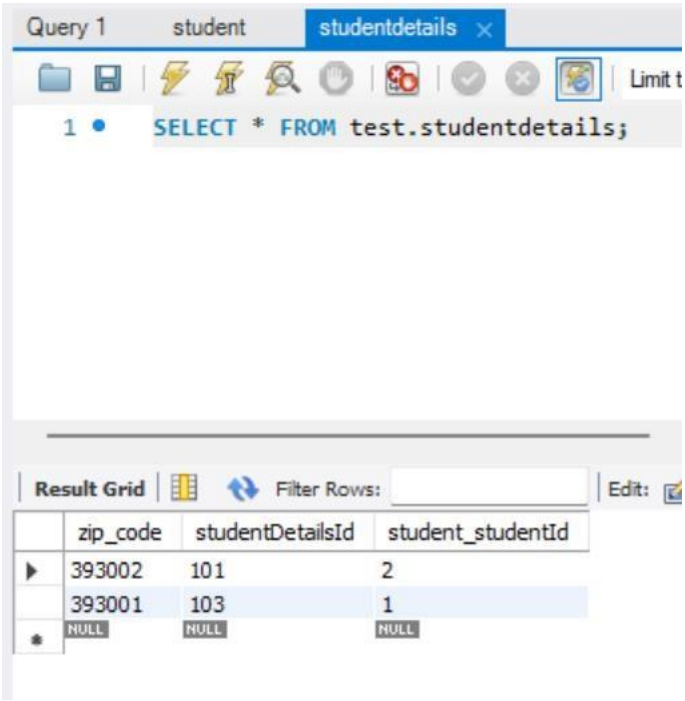
```java
StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);

StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);

Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);

Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);
```

Bidirectional one-to-one mapping:

```java
@Entity
public class StudentDetails {
    @Id
    private Long studentDetailsId;
    2 usages
    @Column(name = "zip_code")
    private int zipCode;
    2 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Student student;

    no usages
    public Student getStudent() {
        return student;
    }

    2 usages
    public void setStudent(Student student) {
        this.student = student;
    }
```



```java
StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);

StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);

Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);

Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);

sd1.setStudent(s2);
sd2.setStudent(s1);
```



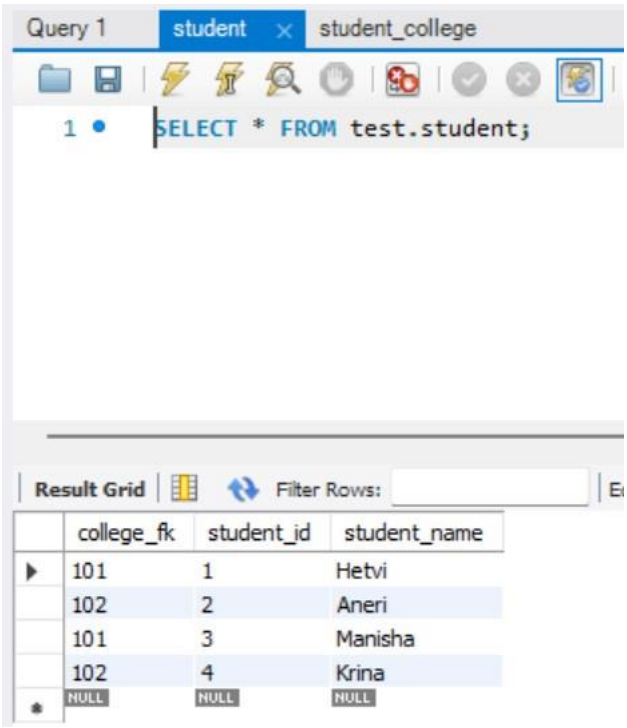
Many-to-one: (Many students are in one college)

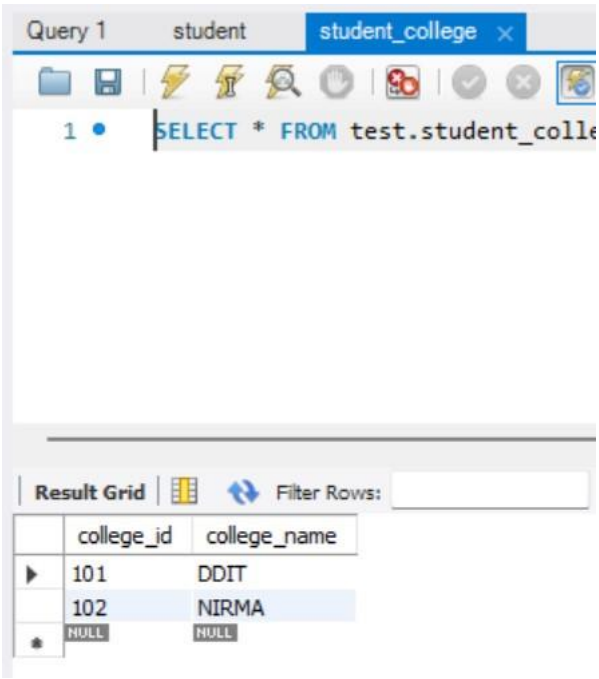
```java
@Entity
public class Student {
    @Id
    @Column(name = "student_id")
    private int studentId;

    2 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "college_fk")
    private Student_College studentCollege;

    no usages
    public Student_College getStudentCollege() { return studentCollege; }

    4 usages
    public void setStudentCollege(Student_College studentCollege) {
        this.studentCollege = studentCollege;
    }
```

One-to-many: (one college have many students)

```java
@Entity
public class Student_College {

    @Id
    @Column(name = "college_id")
    private int collegeId;

    2 usages
    @Column(name = "college_name")
    private String collegeName;

    2 usages
    @OneToMany(mappedBy = "studentCollege",cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Student> students;

    no usages
    public List<Student> getStudents() { return students; }

    2 usages
    public void setStudents(List<Student> students) { this.students = students; }
```

```java
student1.setStudentCollege(college1);
student2.setStudentCollege(college2);
student3.setStudentCollege(college1);
student4.setStudentCollege(college2);

List<Student> ddit_students = new ArrayList<>();
ddit_students.add(student1);
ddit_students.add(student3);

List<Student> nirma_students = new ArrayList<>();
nirma_students.add(student2);
nirma_students.add(student4);

college1.setStudents(ddit_students);
college2.setStudents(nirma_students);
```


Query 1
student
student_college
1 SELECT * FROM test.student;
Result Grid
Filter Rows:
college_fk
student_id
student_name
101
1
Hetvi
102
2
Aneri
101
3
Manisha
102
4
Krina
NULL
NULL
NULL


Query 1
student
student_college
1 SELECT * FROM test.student_colle
Result Grid
Filter Rows:
college_id
college_name
101
DDIT
102
NIRMA
NULL
NULL

Unidirectional Many-to-many: (many students have many certificates)

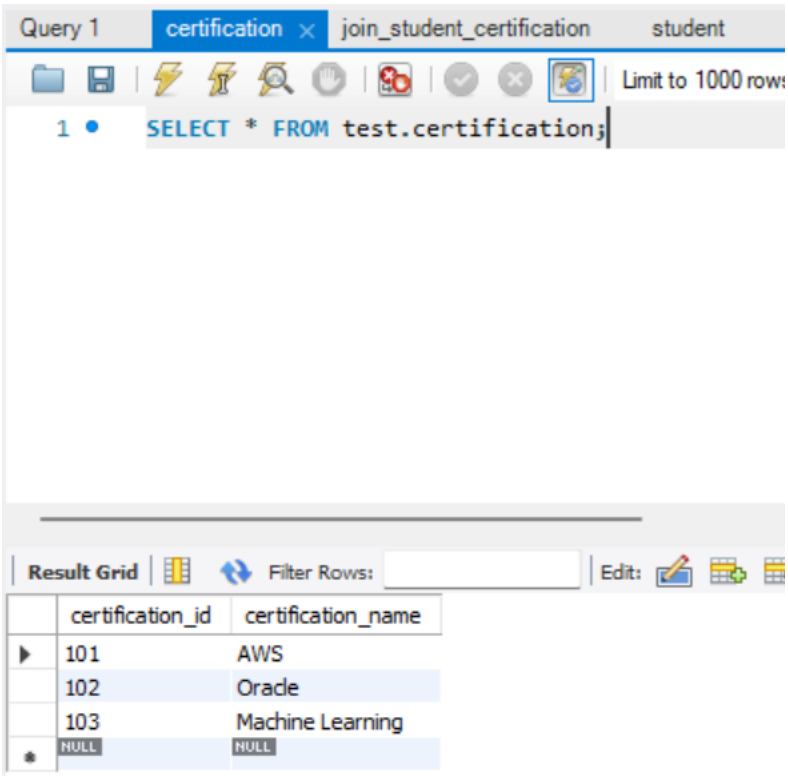
Create certificate entity class with id and name

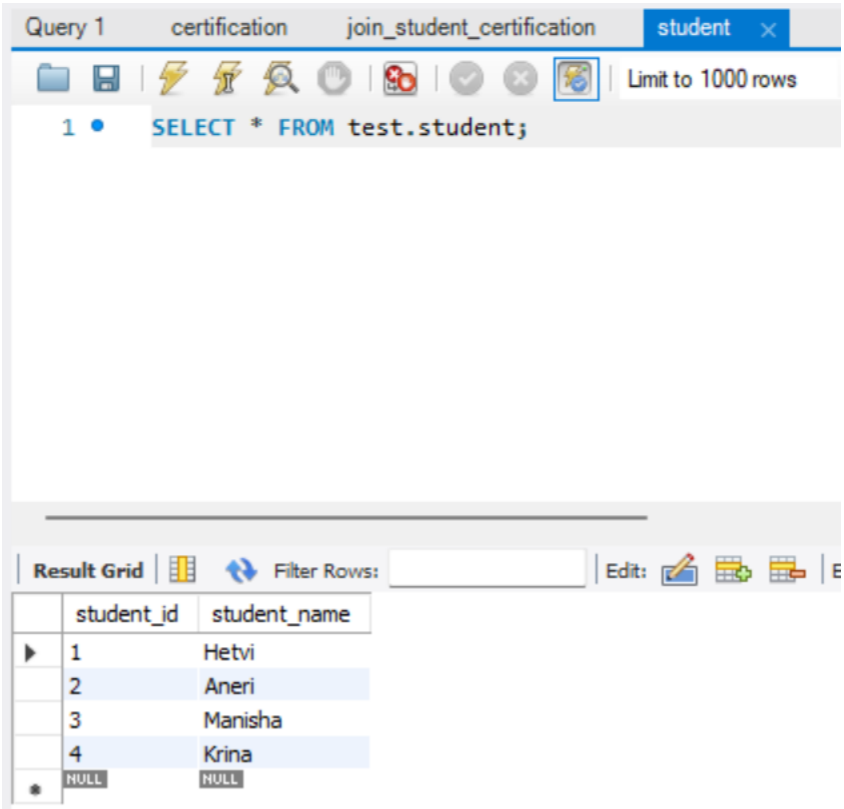
```java
@Entity
public class Student {
    @Id
    @Column(name = "student_id")
    private int studentId;

    2 usages
    @Column(name = "student_name")
    private String studentName;

    2 usages
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "Join_Student_Certification",
            joinColumns = {@JoinColumn(name = "student_id_fk")},
            inverseJoinColumns = {@JoinColumn(name = "certification_id_fk")}
    )
    private Set<Certification> studentCertificates;

    no usages
    public Set<Certification> getStudentCertificates() { return studentCertificates; }
```

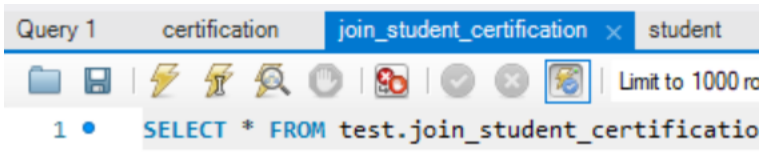
```java
Set<Certification> hetvicerti = new HashSet<>();
hetvicerti.add(machineLearning);
hetvicerti.add(aws);

Set<Certification> aneriCerti = new HashSet<>();
aneriCerti.add(oracle);

Set<Certification> manishaCerti = new HashSet<>();
manishaCerti.add(oracle);
manishaCerti.add(aws);

Set<Certification> krinaCerti = new HashSet<>();
krinaCerti.add(aws);
krinaCerti.add(machineLearning);
krinaCerti.add(oracle);

student1.setStudentCertificates(hetvicerti);
student2.setStudentCertificates(aneriCerti);
student3.setStudentCertificates(manishaCerti);
student4.setStudentCertificates(krinaCerti);
```

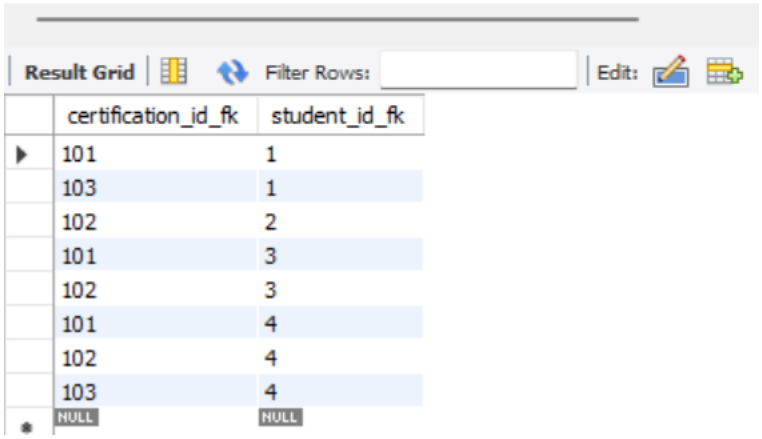
Bidirectional many-to-many mapping:

```java
@Entity
public class Certification {
    @Id
    @Column(name = "certification_id")
    private int certificationId;
    2 usages
    @Column(name = "certification_name")
    private String certificationName;

    2 usages
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "Join_Student_Certification",
            joinColumns = {@JoinColumn(name = "certification_id_fk")},
            inverseJoinColumns = {@JoinColumn(name = "student_id_fk")}
    )
    Set<Student> students;
    no usages
    public Set<Student> getStudents() {
        return students;
    }

    no usages
    public void setStudents(Set<Student> students) {
        this.students = students;
    }
```


Query 1
certification
join_student_certification
student
Limit to 1000 rows
1
SELECT * FROM test.certification;
Result Grid
Filter Rows:
Edit:
certification_id
certification_name
101
AWS
102
Oracle
103
Machine Learning
NULL
NULL


Query 1
certification
join_student_certification
student
Limit to 1000 rows
1
SELECT * FROM test.student;
Result Grid
Filter Rows:
Edit:
student_id
student_name
1
Hetvi
2
Aneri
3
Manisha
4
Krina
NULL
NULL





certification_id_fk	student_id_fk
101	1
103	1
102	2
101	3
102	3
101	4
102	4
103	4
NULL	NULL