1. Method select all student pada kelas StudentMapper.java

```
@Select("select npm, name, gpa from student")
@Results(value = { @Result(property = "npm", column = "npm"),
        @Result(property = "name", column = "name"),
        @Result(property = "gpa", column = "gpa"),
        @Result(property = "courses", column = "npm",
        javaType = List.class,
        many = @Many(select = "selectCourses")) })
List<StudentModel> selectAllStudents();
```

Pada method diatas, anotasi @Result akan memetakan setiap property ke kelas model sesuai dengan nama variabelnya dan column akan diisi sesuai dengan hasil kueri ke database eaap.

```
        @Result(property = "courses", column = "npm",
        javaType = List.class,
        many = @Many(select = "selectCourses")) })
List<StudentModel> selectAllStudents();
```

Untuk @Result yang terakhir, property diset dengan variabel courses sesuai pada kelas model StudentModel.java yang akan diisi oleh variabel many yang diisi oleh method selectCourses, column diisi dengan npm karena akan dijadikan parameter yang dikirimkan ke method selectCourses. Variabel javaType berisi kelas yang akan dikembalikan, dalam hal ini karena kita akan mengisi dengan list courses yang diambil student maka kembaliannya akan berisi List.

2. View pada halaman http://localhost:8080/course/view/{id} untuk Course

```
@RequestMapping("/course/view/{id_course}")
public String viewCourse (Model model,
        @PathVariable(value = "id_course") String id)
{
    CourseModel course = studentDAO.selectCourse(id);

    if (course != null) {
        model.addAttribute ("course", course);
        return "viewcourse";
    } else {
        model.addAttribute ("id_course", id);
        return "not-found-course";
    }
}
```

Pada method viewCourse di kelas controller, program akan memanggil method selectCourse(id) pada kelas StudentService.java yang berisikan parameter id course yang dimasukkan pada url.

```
@Override
public CourseModel selectCourse(String id) {
     log.info ("select course with id_course {}", id);
     return studentMapper.selectCourse (id);
```

Kemudian pada kelas StudentService.java akan dilakukan pemanggilan ke kelas mapper dengan parameter id course.

```
@Select("select id_course, name, credits from course where id_course = #{id}")
@Results(value = { @Result(property = "idCourse", column = "id_course"),
        @Result(property = "name", column = "name"),
        @Result(property = "credits", column = "credits"),
        @Result(property = "students", column = "id_course",
        javaType = List.class,
        many = @Many(select = "selectStudentClass")) })
CourseModel selectCourse(@Param("id") String id);
```

Pada kelas mapper akan diambil data id_course, nama dan credits dari course sesuai id yang dikirimkan. Untuk @Result yang terakhir, property diset dengan variabel students sesuai pada kelas model CourseModel.java yang akan diisi oleh variabel many yang diisi oleh method selectStudentClass, column diisi dengan id course karena akan dijadikan parameter yang dikirimkan ke method selectStudentClass. Variabel javaType berisi kelas yang akan dikembalikan, dalam hal ini karena kita akan mengisi dengan list student yang mengambil sebuah course maka kembaliannya akan berisi List.

apabila hasil query bernilai false maka akan dikirim view ke not-found-course.html dengan message “Course not found” beserta id coursenya, namun apabila variabel course tidak null, maka akan dikembalikan viewcourse.html yang berisi nama course, credits dan daftar peserta course.