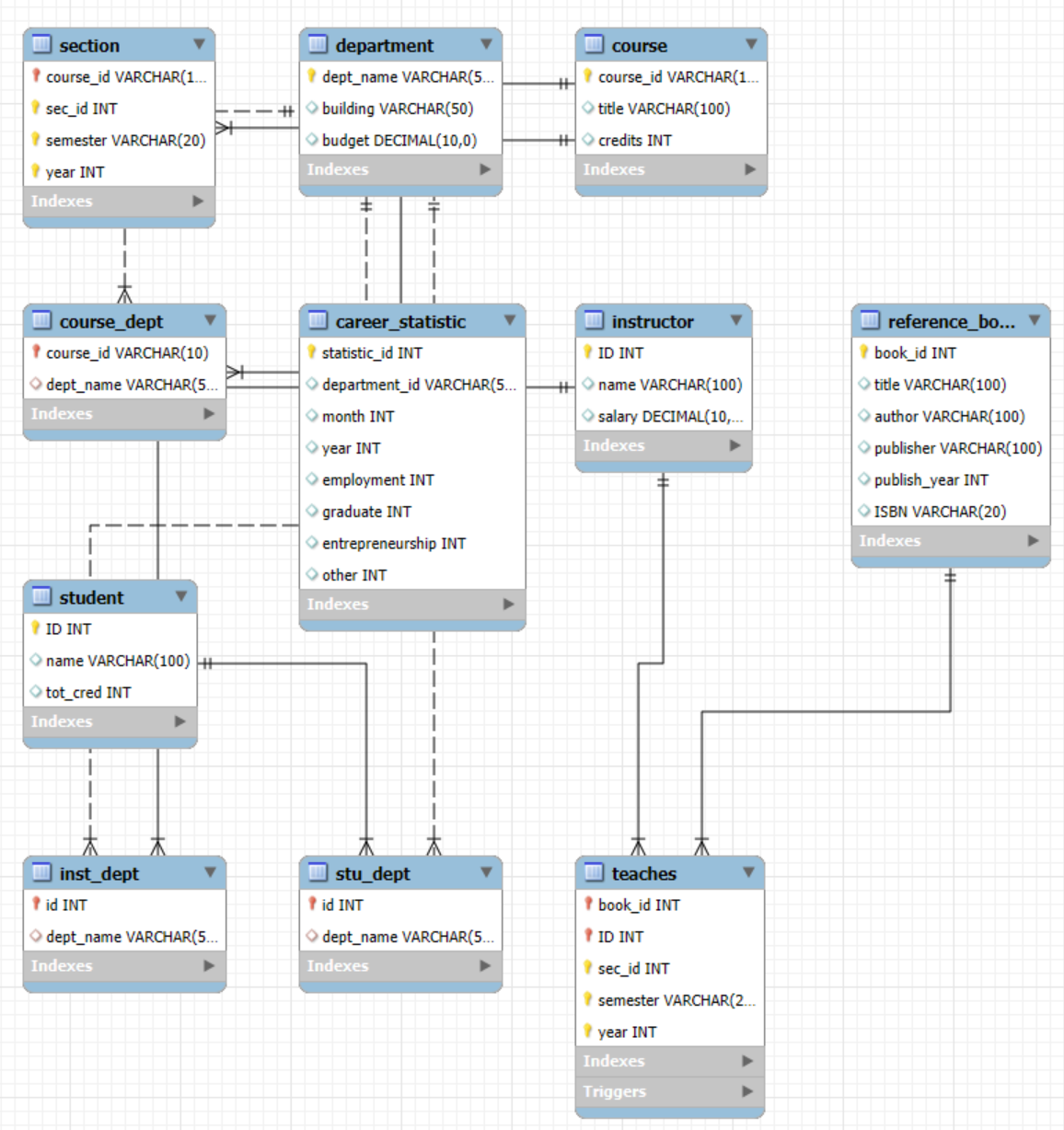
**1. 12주차 과제의 최종 결과로 제출한 RDB 스키마 다이어그램**

****

**2. CREATE TABLE문: DB에 생성한 테이블 별로 해당 CREATE TABLE문**

#### ****1. Department 테이블****

* 학과 정보를 저장하는 테이블. 학과 이름(dept\_name)을 기본 키로 사용하며, 건물 위치와 예산 정보를 포함합니다.

CREATE TABLE Department (

dept\_name VARCHAR(50) PRIMARY KEY, -- 학과 이름

building VARCHAR(50), -- 학과 건물 위치

budget DECIMAL -- 학과 예산

)ENGINE=InnoDB;

#### ****2. Instructor 테이블****

* 교수 정보를 저장하는 테이블. 교수의 ID(ID)를 기본 키로 사용하며, 이름과 급여 정보를 포함합니다.

CREATE TABLE Instructor (

ID INT PRIMARY KEY, -- 교수 ID

name VARCHAR(100), -- 교수 이름

salary DECIMAL -- 급여

)ENGINE=InnoDB;

#### ****3. Student 테이블****

* 학생 정보를 저장하는 테이블. 학생 ID(ID)를 기본 키로 사용하며, 이름과 이수 학점 정보를 포함합니다.

CREATE TABLE Student (

ID INT PRIMARY KEY, -- 학생 ID

name VARCHAR(100), -- 학생 이름

tot\_cred INT -- 이수 학점

)ENGINE=InnoDB;

#### ****4. Course 테이블****

* 과목 정보를 저장하는 테이블. 과목 ID(course\_id)를 기본 키로 사용하며, 과목 제목과 학점 정보를 포함합니다.

CREATE TABLE Course (

course\_id VARCHAR(10) PRIMARY KEY, -- 과목 ID

title VARCHAR(100), -- 과목 제목

credits INT -- 학점

)ENGINE=InnoDB;

#### ****5. Section 테이블****

* 수업 정보를 저장하는 테이블. 과목 ID와 수업 ID(sec\_id), 학기(semester), 연도(year)의 복합 키로 유일성을 보장합니다.

CREATE TABLE Section (

course\_id VARCHAR(10), -- 과목 ID

sec\_id INT, -- 수업 ID

semester VARCHAR(20), -- 학기

year INT, -- 연도

PRIMARY KEY (course\_id, sec\_id, semester, year), -- 복합 기본 키

FOREIGN KEY (course\_id) REFERENCES Course(course\_id) -- 외래 키: Course 참조

)ENGINE=InnoDB;

#### ****6. Reference\_book 테이블****

* 참고 도서 정보를 저장하는 테이블. 도서 ID(book\_id)를 기본 키로 사용하며, 제목, 저자, 출판사, 출판 연도, ISBN을 포함합니다.

CREATE TABLE Reference\_book (

book\_id INT PRIMARY KEY, -- 도서 ID

title VARCHAR(100), -- 도서 제목

author VARCHAR(100), -- 저자

publisher VARCHAR(100), -- 출판사

publish\_year INT, -- 출판 연도

ISBN VARCHAR(20) -- ISBN

)ENGINE=InnoDB;

#### ****7. Teaches 테이블****

* 교수, 수업, 참고 도서 간의 관계를 저장하는 테이블. 복합 키(book\_id, ID, sec\_id, semester, year)로 유일성을 보장합니다.

CREATE TABLE Teaches (

book\_id INT, -- 도서 ID

ID INT, -- 교수 ID

sec\_id INT, -- 수업 ID

semester VARCHAR(20), -- 학기

year INT, -- 연도

PRIMARY KEY (book\_id, ID, sec\_id, semester, year), -- 복합 기본 키

FOREIGN KEY (book\_id) REFERENCES Reference\_book(book\_id), -- 외래 키: Reference\_book 참조

FOREIGN KEY (ID) REFERENCES Instructor(ID) -- 외래 키: Instructor 참조

)ENGINE=InnoDB;

#### ****8. Career\_statistic 테이블****

* 학과별 진로 통계 정보를 저장하는 테이블. 통계 ID(statistic\_id)를 기본 키로 사용하며, 월별, 연도별 취업, 졸업, 창업, 기타 데이터를 저장합니다.

CREATE TABLE Career\_statistic (

statistic\_id INT PRIMARY KEY, -- 통계 ID

department\_id VARCHAR(50), -- 학과 ID

month INT, -- 월

year INT, -- 연도

employment INT, -- 취업자 수

graduate INT, -- 졸업자 수

entrepreneurship INT, -- 창업자 수

other INT -- 기타

)ENGINE=InnoDB;

#### ****9. Stu\_dept 테이블****

* 학생과 학과 간의 관계를 저장하는 테이블. 학생 ID(id)를 기본 키로 사용하며, 학과 이름(dept\_name)을 참조합니다.

CREATE TABLE Stu\_dept (

id INT PRIMARY KEY, -- 학생 ID

dept\_name VARCHAR(50), -- 학과 이름

FOREIGN KEY (id) REFERENCES Student(ID), -- 외래 키: Student 참조

FOREIGN KEY (dept\_name) REFERENCES Department(dept\_name) -- 외래 키: Department 참조

)ENGINE=InnoDB;

#### ****10. Inst\_dept 테이블****

* 교수와 학과 간의 관계를 저장하는 테이블. 교수 ID(id)를 기본 키로 사용하며, 학과 이름(dept\_name)을 참조합니다.

CREATE TABLE Inst\_dept (

id INT PRIMARY KEY, -- 교수 ID

dept\_name VARCHAR(50), -- 학과 이름

FOREIGN KEY (id) REFERENCES Instructor(ID), -- 외래 키: Instructor 참조

FOREIGN KEY (dept\_name) REFERENCES Department(dept\_name) -- 외래 키: Department 참조

)ENGINE=InnoDB;

#### ****11. Course\_dept 테이블****

* 과목과 학과 간의 관계를 저장하는 테이블. 과목 ID(course\_id)를 기본 키로 사용하며, 학과 이름(dept\_name)을 참조합니다.

CREATE TABLE Course\_dept (

course\_id VARCHAR(10) PRIMARY KEY, -- 과목 ID

dept\_name VARCHAR(50), -- 학과 이름

FOREIGN KEY (course\_id) REFERENCES Course(course\_id), -- 외래 키: Course 참조

FOREIGN KEY (dept\_name) REFERENCES Department(dept\_name) -- 외래 키: Department 참조

)ENGINE=InnoDB;

**3. SQL문: Java 프로그램에서 사용한 SQL문들을 모두 쓰고 간략히 설명**

1. **BookReferenceDAO에서 사용된 쿼리**

SELECT t.ID AS professor\_id, rb.book\_id, rb.title, rb.author, rb.publisher, rb.isbn, s.course\_id

FROM Teaches t

JOIN Reference\_book rb ON t.book\_id = rb.book\_id

JOIN Section s ON t.sec\_id = s.sec\_id AND t.semester = s.semester AND t.year = s.year

WHERE s.course\_id = ?

AND t.sec\_id = ?

AND t.semester = ?

AND t.year = ?;

* + 특정 수업(course\_id, sec\_id, semester, year)에 대해 담당 교수와 참고 도서 정보를 조회합니다.

1. **BookLoanDAO에서 사용된 쿼리**

SELECT cd.dept\_name

FROM Reference\_book rb

JOIN Teaches t ON rb.book\_id = t.book\_id

JOIN Section s ON t.sec\_id = s.sec\_id AND t.semester = s.semester AND t.year = s.year

JOIN Course\_Dept cd ON s.course\_id = cd.course\_id

WHERE rb.ISBN = ?;

* + 특정 ISBN에 해당하는 도서가 어느 학과의 수업에 지정되었는지 조회합니다.

1. **CareerStatisticDAO에서 사용된 쿼리**

SELECT \*

FROM Career\_statistic

WHERE department\_id = ?

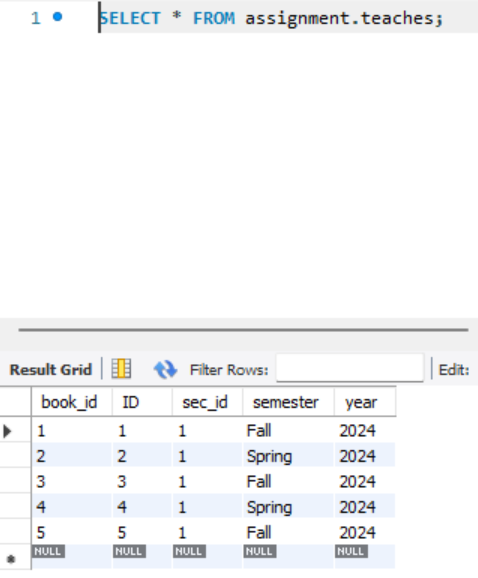
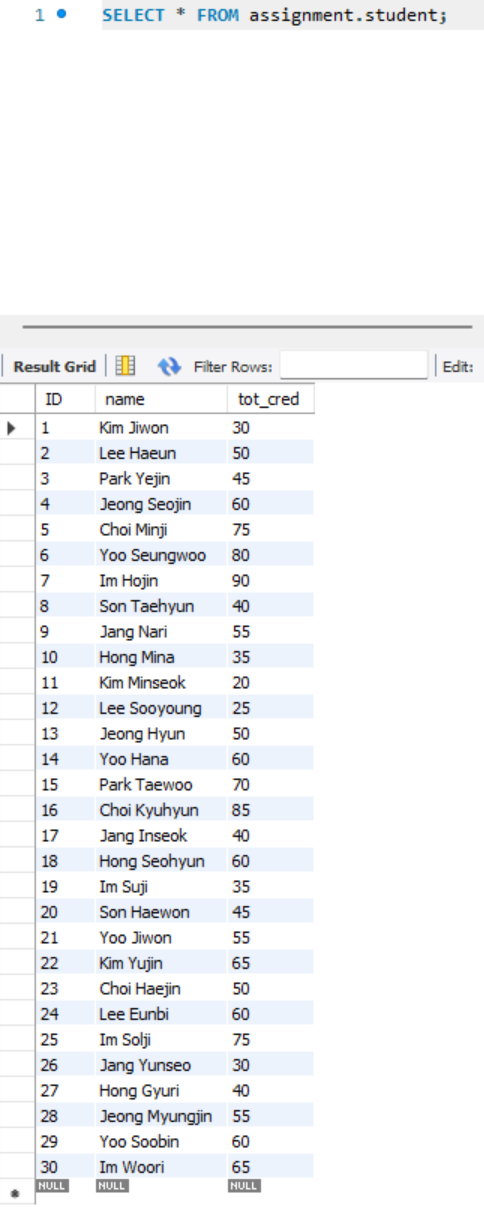
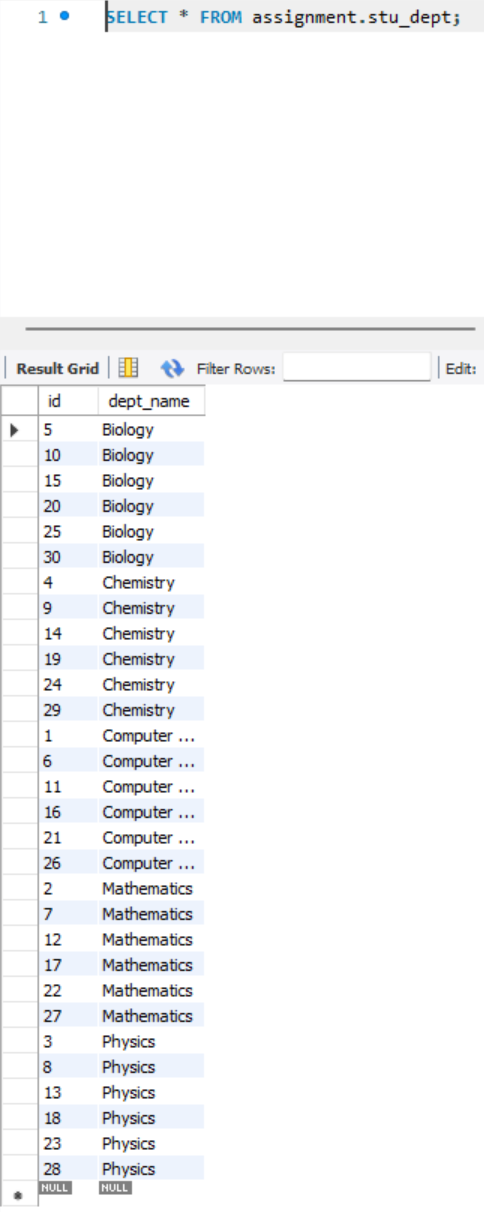
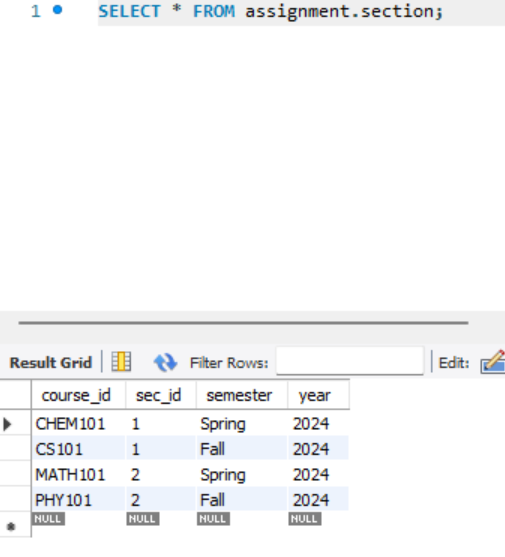
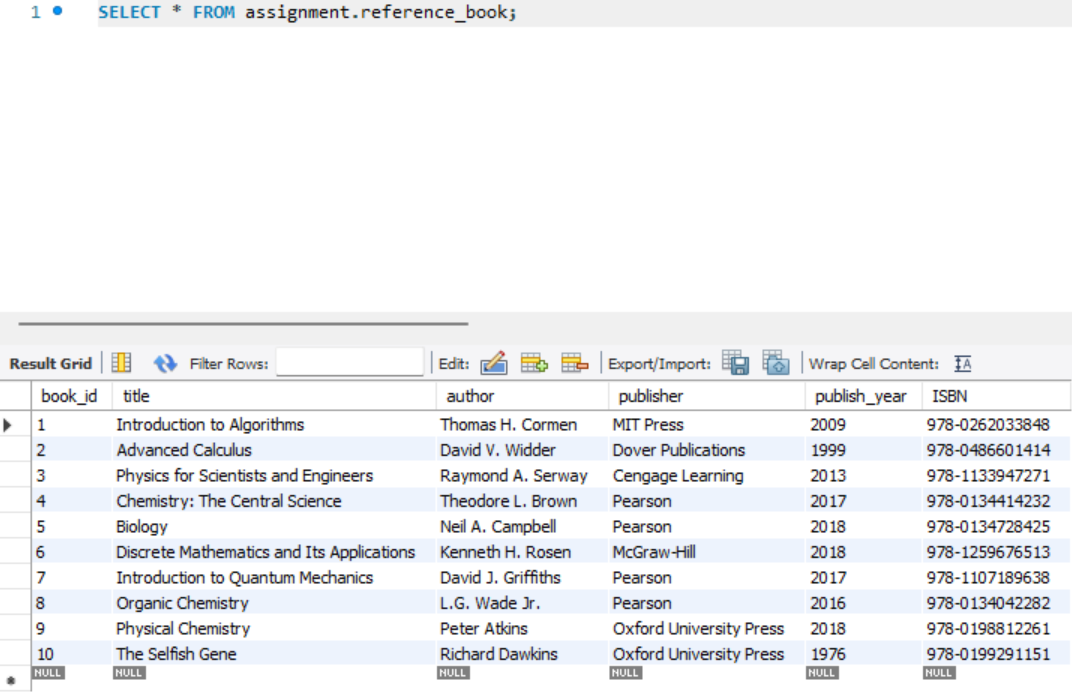
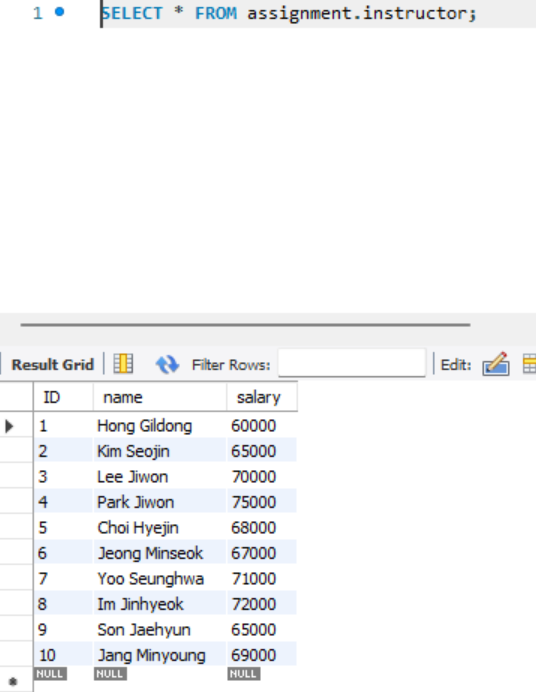
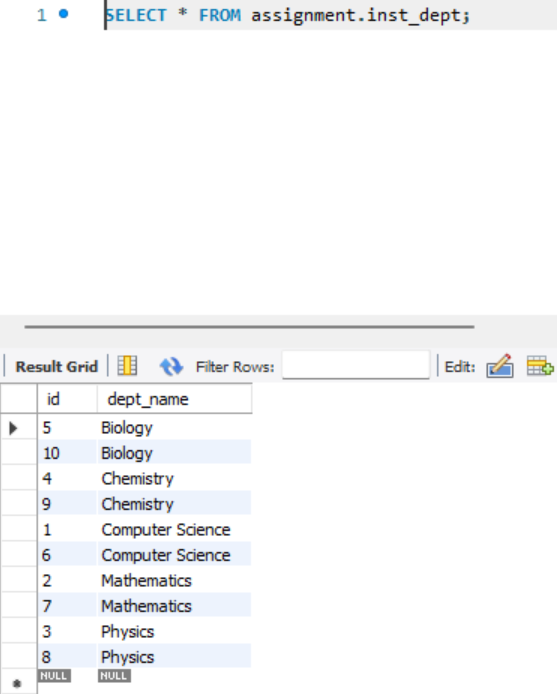
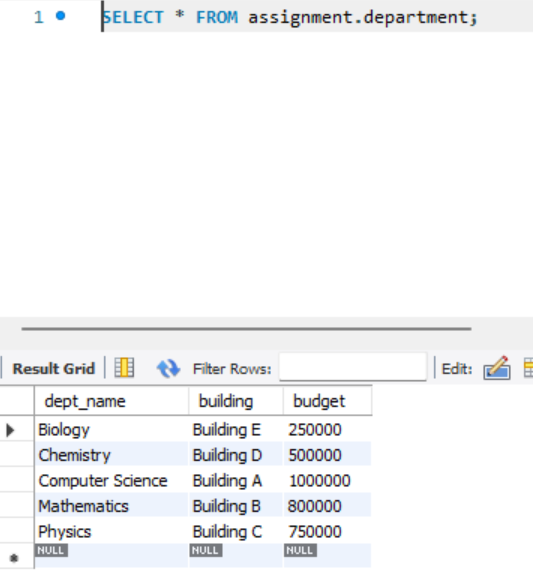
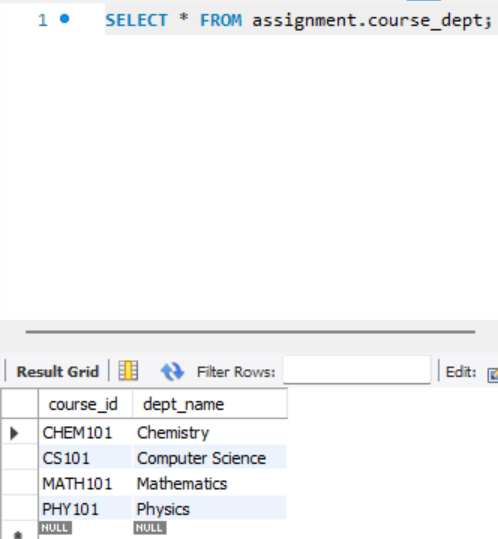
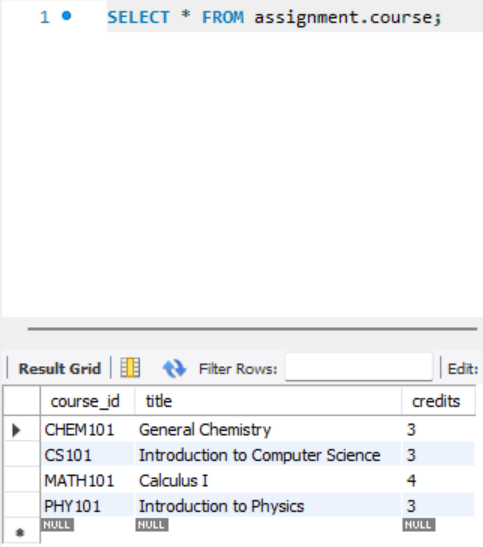
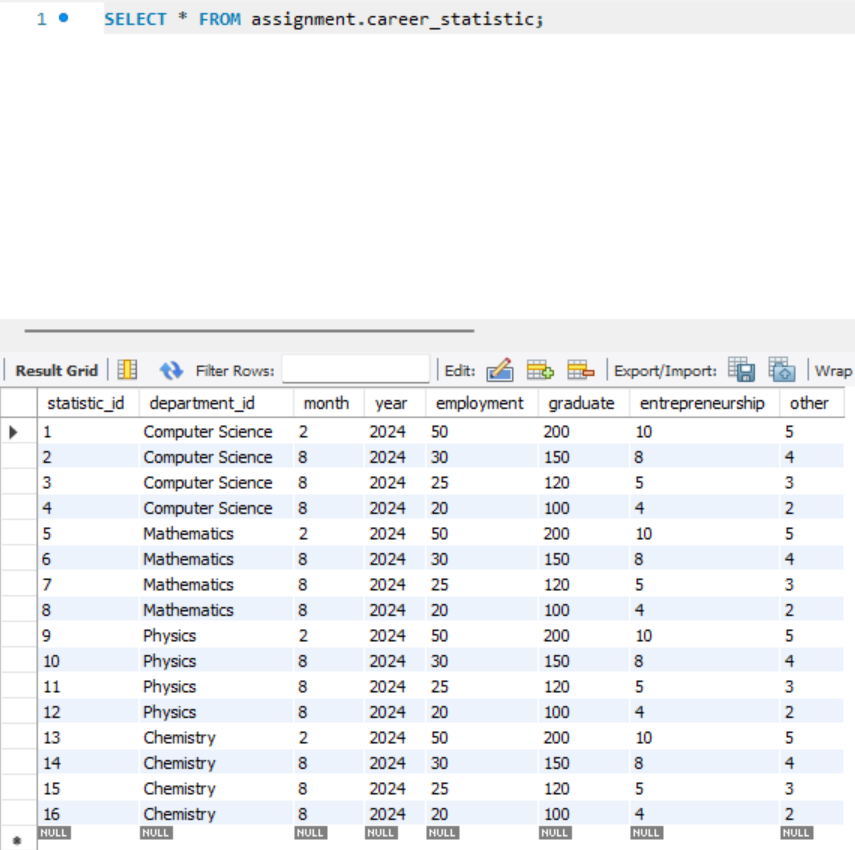
AND year = ?

AND month = ?;

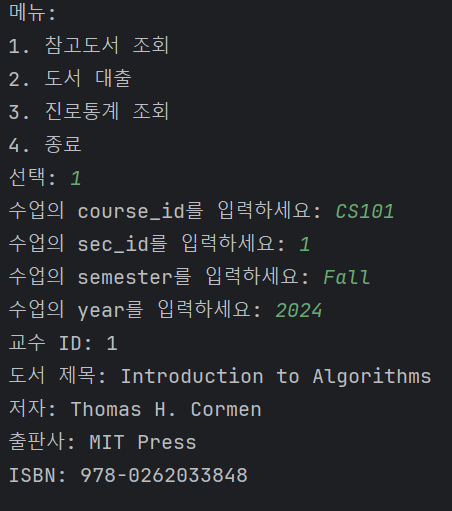
* + 특정 학과, 연도, 월의 진로 통계를 조회합니다.

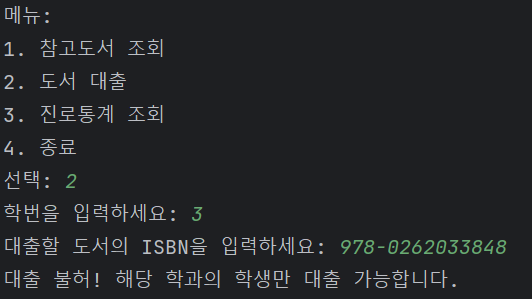
**4. 프로그램 로직: 기능 메뉴 1번~3번의 전체 로직 요약**

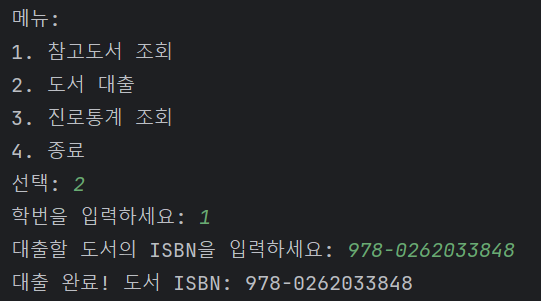
* **1번: 참고 도서 조회**
  + **입력**: 사용자가 course\_id, sec\_id, semester, year를 입력.
  + **처리**:
    - BookReferenceDAO.getReferenceBooksForCourse 호출.
    - 해당 수업에 지정된 참고 도서와 담당 교수 정보를 조회.
    - 교수의 학과와 수업의 학과를 비교하여 일치하면 결과 출력.
  + **출력**: 일치하는 교수 ID와 참고 도서 정보.
* **2번: 도서 대출**
  + **입력**: 사용자가 학번과 대출할 도서의 ISBN 입력.
  + **처리**:
    - BookLoanDAO.borrowBook 호출.
    - 도서의 학과와 학생의 학과를 비교.
    - 학과가 일치하면 대출 완료 메시지 출력, 불일치하면 대출 불가 메시지 출력.
  + **출력**: 대출 가능 여부 메시지.
* **3번: 진로 통계 조회**
  + **입력**: 학과명과 연도 범위를 입력.
  + **처리**:
    - CareerStatisticDAO.getCareerStatistics 호출.
    - 연도별 및 월별 진로 통계를 가져와 출력.
  + **출력**: 진로 통계 정보.

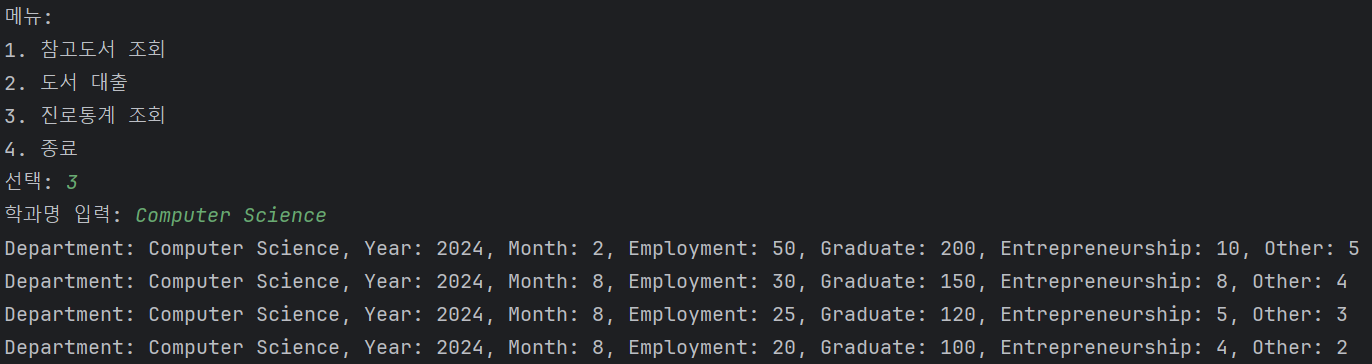
**5. DB 데이터: 테이블의 모든 튜플 출력**

**6. Java 프로그램 실행 결과**









**7. 정확성 검증**

### ****1. 참고 도서 조회 검증****

#### ****입력값****

course\_id: CS101

sec\_id: 1

semester: Fall

year: 2024

#### ****출력값****

교수 ID: 1

도서 제목: Introduction to Algorithms

저자: Thomas H. Cormen

출판사: MIT Press

ISBN: 978-0262033848

### ****데이터베이스 데이터****

1. **Reference\_book 테이블**

book\_id | title | author | publisher | ISBN

--------------------------------------------------------------------------------------

1 | Introduction to Algorithms | Thomas H. Cormen | MIT Press | 978-0262033848

1. **Teaches 테이블**

book\_id | ID | sec\_id | semester | year

----------------------------------------

1 | 1 | 1 | Fall | 2024

1. **Section 테이블**

course\_id | sec\_id | semester | year

------------------------------------

CS101 | 1 | Fall | 2024

1. **Inst\_Dept 테이블**

id | dept\_name

-----------------------

1 | Computer Science

1. **Course\_Dept 테이블**

course\_id | dept\_name

---------------------

CS101 | Computer Science

### ****검증 결과****

* **교수 ID**:
  + Teaches 테이블에서 CS101(course\_id)과 1(sec\_id), Fall(semester), 2024(year)에 해당하는 교수는 ID=1로 저장되어 있습니다.
  + **출력값**: 교수 ID: 1로 정확히 출력되었습니다.
* **도서 정보**:
  + Reference\_book 테이블에서 book\_id=1인 도서는 Introduction to Algorithms입니다.
  + **출력값**:

도서 제목: Introduction to Algorithms

저자: Thomas H. Cormen

출판사: MIT Press

ISBN: 978-0262033848

* + - 테이블의 데이터와 정확히 일치합니다.
* **학과 검증**:
  + Inst\_Dept 테이블에서 교수 ID 1의 학과는 Computer Science입니다.
  + Course\_Dept 테이블에서 CS101 수업의 학과는 Computer Science입니다.
  + 교수의 학과와 수업의 학과가 동일하므로 결과 출력 조건이 충족되었습니다.

### ****2. 도서 대출 검증****

#### ****첫 번째 입력****

* **입력값**:

학번: 3

ISBN: 978-0262033848

* **출력값**:

대출 불허! 해당 학과의 학생만 대출 가능합니다.

* **데이터베이스 데이터**:
  + Reference\_book 테이블:

book\_id | title | ISBN | ...

1 | Introduction to Algorithms | 978-0262033848 | ...

* + Teaches 테이블:

book\_id | ID | sec\_id | semester | year

1 | 1 | 1 | Fall | 2024

* + Section 테이블:

course\_id | sec\_id | semester | year

CS101 | 1 | Fall | 2024

* + Course\_Dept 테이블:

course\_id | dept\_name

CS101 | Computer Science

* + Stu\_Dept 테이블:

id | dept\_name

3 | Physics

* **검증 결과**:
  + 도서 978-0262033848는 CS101 수업(학과: Computer Science)에 지정된 도서입니다.
  + 학번 3의 학생은 Physics 학과 소속으로, 도서의 학과와 일치하지 않아 대출이 불허되었습니다.

#### ****두 번째 입력****

* **입력값**:

학번: 1

ISBN: 978-0262033848

* **출력값**:

대출 완료! 도서 ISBN: 978-0262033848

* **데이터베이스 데이터**:
  + 동일한 도서 데이터(Reference\_book, Teaches, Section, Course\_Dept).
  + Stu\_Dept 테이블:

id | dept\_name

1 | Computer Science

* **검증 결과**:
  + 도서 978-0262033848는 CS101 수업(학과: Computer Science)에 지정된 도서입니다.
  + 학번 1의 학생은 Computer Science 학과 소속으로, 도서의 학과와 일치하여 대출이 허용되었습니다.

### ****3. 진로 통계 조회 검증****

#### ****입력값****

학과명 입력: Computer Science

#### ****출력값****

Department: Computer Science, Year: 2024, Month: 2, Employment: 50, Graduate: 200, Entrepreneurship: 10, Other: 5

Department: Computer Science, Year: 2024, Month: 8, Employment: 30, Graduate: 150, Entrepreneurship: 8, Other: 4

Department: Computer Science, Year: 2024, Month: 8, Employment: 25, Graduate: 120, Entrepreneurship: 5, Other: 3

Department: Computer Science, Year: 2024, Month: 8, Employment: 20, Graduate: 100, Entrepreneurship: 4, Other: 2

#### ****데이터베이스 데이터****

* Career\_statistic 테이블:

statistic\_id | department\_id | month | year | employment | graduate | entrepreneurship | other

1 | Computer Science | 2 | 2024 | 50 | 200 | 10 | 5

2 | Computer Science | 8 | 2024 | 30 | 150 | 8 | 4

3 | Computer Science | 8 | 2024 | 25 | 120 | 5 | 3

4 | Computer Science | 8 | 2024 | 20 | 100 | 4 | 2

#### ****검증 결과****

* 프로그램이 조회한 데이터는 Career\_statistic 테이블과 정확히 일치합니다.
* 출력값이 입력한 학과와 연도, 월별 데이터를 정확히 반환하며, 모든 필드(Employment, Graduate, Entrepreneurship, Other)의 값이 DB와 일치합니다.
* **8. Java/JDBC 프로그램 소스코드**
* **Main.java**

import java.util.List;

import java.util.Scanner;

import DB.DAO.\*;

import DB.Models.CareerStatistic;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

while (true) {

System.out.println("메뉴:");

System.out.println("1. 참고도서 조회");

System.out.println("2. 도서 대출");

System.out.println("3. 진로통계 조회");

System.out.println("4. 종료");

System.out.print("선택: ");

choice = scanner.nextInt();

if (choice == 1) {

System.out.print("수업의 course\_id를 입력하세요: ");

String courseId = scanner.next();

System.out.print("수업의 sec\_id를 입력하세요: ");

int secId = scanner.nextInt();

System.out.print("수업의 semester를 입력하세요: ");

String semester = scanner.next();

System.out.print("수업의 year를 입력하세요: ");

int year = scanner.nextInt();

BookReferenceDAO.getReferenceBooksForCourse(courseId, secId, semester, year);

} else if (choice == 2) {

System.out.print("학번을 입력하세요: ");

int studentId = scanner.nextInt();

System.out.print("대출할 도서의 ISBN을 입력하세요: ");

String bookIsbn = scanner.next();

BookLoanDAO.borrowBook(studentId, bookIsbn);

} else if (choice == 3) {

System.out.print("학과명 입력: ");

scanner.nextLine(); // 개행 문자 제거

String departmentId = scanner.nextLine();

for (int year = 2020; year < 2025; year++) {

List<CareerStatistic> stats2 = CareerStatisticDAO.getCareerStatistics(departmentId, year, 2);

stats2.forEach(System.out::println);

List<CareerStatistic> stats8 = CareerStatisticDAO.getCareerStatistics(departmentId, year, 8);

stats8.forEach(System.out::println);

}

} else if (choice == 4) {

System.out.println("종료");

break;

}

}

}

}

* **DBConnection.java**

package DB;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnection {

private static final String URL = "jdbc:mysql://localhost:3306/assignment";

private static final String USER = "root";

private static final String PASSWORD = "root";

public static Connection getConnection() throws SQLException {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

return DriverManager.getConnection(URL, USER, PASSWORD);

} catch (ClassNotFoundException e) {

throw new SQLException("MySQL JDBC Driver not found.", e);

}

}

}

* **BookReferenceDAO.java**

package DB.DAO;

import java.sql.\*;

import java.util.HashMap;

import java.util.Map;

import DB.DBConnection;

public class BookReferenceDAO {

public static void getReferenceBooksForCourse(String courseId, int secId, String semester, int year) {

String query =

"SELECT t.ID AS professor\_id, rb.book\_id, rb.title, rb.author, rb.publisher, rb.isbn, s.course\_id " +

"FROM Teaches t " +

"JOIN Reference\_book rb ON t.book\_id = rb.book\_id " +

"JOIN Section s ON t.sec\_id = s.sec\_id AND t.semester = s.semester AND t.year = s.year " +

"WHERE s.course\_id = ? " +

"AND t.sec\_id = ? " +

"AND t.semester = ? " +

"AND t.year = ?";

try (Connection conn = DBConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, courseId);

stmt.setInt(2, secId);

stmt.setString(3, semester);

stmt.setInt(4, year);

ResultSet rs = stmt.executeQuery();

Map<Integer, Map<Integer, String>> professorBookMap = new HashMap<>();

while (rs.next()) {

int professorId = rs.getInt("professor\_id");

int bookId = rs.getInt("book\_id");

String title = rs.getString("title");

String author = rs.getString("author");

String publisher = rs.getString("publisher");

String isbn = rs.getString("isbn");

professorBookMap.computeIfAbsent(professorId, k -> new HashMap<>())

.put(bookId, String.format("도서 제목: %s\n저자: %s\n출판사: %s\nISBN: %s",

title, author, publisher, isbn));

}

for (int professorId : professorBookMap.keySet()) {

String deptQuery =

"SELECT d.dept\_name " +

"FROM Inst\_Dept idp " +

"JOIN Department d ON idp.dept\_name = d.dept\_name " +

"WHERE idp.id = ?";

try (PreparedStatement deptStmt = conn.prepareStatement(deptQuery)) {

deptStmt.setInt(1, professorId);

ResultSet deptRs = deptStmt.executeQuery();

if (deptRs.next()) {

String professorDept = deptRs.getString("dept\_name");

String courseDeptQuery =

"SELECT cd.dept\_name " +

"FROM Course\_Dept cd " +

"WHERE cd.course\_id = ?";

try (PreparedStatement courseDeptStmt = conn.prepareStatement(courseDeptQuery)) {

courseDeptStmt.setString(1, courseId);

ResultSet courseDeptRs = courseDeptStmt.executeQuery();

if (courseDeptRs.next()) {

String courseDept = courseDeptRs.getString("dept\_name");

if (professorDept.equals(courseDept)) {

System.out.println("교수 ID: " + professorId);

Map<Integer, String> books = professorBookMap.get(professorId);

for (String bookDetails : books.values()) {

System.out.println(bookDetails + "\n");

}

}

}

}

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

* **BookLoanDAO.java**

package DB.DAO;

import java.sql.\*;

import DB.DBConnection;

public class BookLoanDAO {

public static void borrowBook(int studentId, String bookIsbn) {

String findCourseDeptQuery =

"SELECT cd.dept\_name FROM Reference\_book rb JOIN Teaches t ON rb.book\_id = t.book\_id " +

"JOIN Section s ON t.sec\_id = s.sec\_id AND t.semester = s.semester AND t.year = s.year " +

"JOIN Course\_Dept cd ON s.course\_id = cd.course\_id WHERE rb.ISBN = ?";

String findStudentDeptQuery =

"SELECT sd.dept\_name " +

"FROM Stu\_dept sd " +

"WHERE sd.id = ?";

try (Connection conn = DBConnection.getConnection();

PreparedStatement courseDeptStmt = conn.prepareStatement(findCourseDeptQuery);

PreparedStatement studentDeptStmt = conn.prepareStatement(findStudentDeptQuery)) {

courseDeptStmt.setString(1, bookIsbn);

ResultSet courseDeptRs = courseDeptStmt.executeQuery();

if (courseDeptRs.next()) {

String courseDept = courseDeptRs.getString("dept\_name");

studentDeptStmt.setInt(1, studentId);

ResultSet studentDeptRs = studentDeptStmt.executeQuery();

if (studentDeptRs.next()) {

String studentDept = studentDeptRs.getString("dept\_name");

if (courseDept.equals(studentDept)) {

System.out.println("대출 완료! 도서 ISBN: " + bookIsbn);

} else {

System.out.println("대출 불허! 해당 학과의 학생만 대출 가능합니다.");

}

}

} else {

System.out.println("대출 불허! 해당 도서는 수업에 지정된 도서가 아닙니다.");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

* **CareerStatisticDAO.java**

package DB.DAO;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import DB.DBConnection;

import DB.Models.CareerStatistic;

public class CareerStatisticDAO {

public static List<CareerStatistic> getCareerStatistics(String departmentId, int year, int month) {

List<CareerStatistic> stats = new ArrayList<>();

String query = "SELECT \* FROM Career\_statistic WHERE department\_id = ? AND year = ? AND month = ?";

try (Connection conn = DBConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, departmentId);

stmt.setInt(2, year);

stmt.setInt(3, month);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

int statisticId = rs.getInt("statistic\_id");

int employment = rs.getInt("employment");

int graduate = rs.getInt("graduate");

int entrepreneurship = rs.getInt("entrepreneurship");

int other = rs.getInt("other");

stats.add(new CareerStatistic(statisticId, departmentId, month, year, employment, graduate, entrepreneurship, other));

}

} catch (SQLException e) {

e.printStackTrace();

}

return stats;

}

}