REPORT

“ 데이터베이스시스템 3차 과제 ”

과 목 명	데이터베이스시스템
담당교수	손용락 교수님
학 과	컴퓨터공학과
학 번	2016305078
이 름	최영환
제 출 일	2021.06.09

1. 교수 ID를 입력받은 후 해당 교수가 맡아 진행한(teach) 강좌의 교과목명(title)을 검색하는 프로그램을 작성하시오. 해당 교수가 한 번도 강좌를 맡은 적이 없을 경우 "맡은 강좌가 없음"을 출력하도록 하시오.

< 전체 소스 코드 >

```
import java.sql.*;

public class report_01 {
    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                                 + "database=largeDB;integratedSecurity=true";
            Connection conn = DriverManager.getConnection(connectionUrl);
            Statement stmt = conn.createStatement();
            System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
            Scanner sc = new Scanner(System.in);

            System.out.print("교수의 ID를 입력하세요 : ");
            String i_ID = sc.nextLine();
            ResultSet rs = stmt.executeQuery("SELECT * FROM teaches, course "
                                           + "WHERE teaches.ID = " + i_ID
                                           + "and teaches.course_id = course.course_id");

            System.out.println("ID가 " + i_ID + "인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...");

            if (rs.next()) {
                System.out.println("해당 교수가 맡아서 진행한 강좌의 교과목명은 다음과 같습니다.");
                do {
                    String field1 = rs.getString("title");
                    System.out.println(field1);
                } while(rs.next());
            }
            else {
                System.out.println("해당 교수는 맡은 강좌가 없음.");
            }

            sc.close();
            rs.close();
            stmt.close();
            conn.close();
            } catch (ClassNotFoundException sqle) {
            System.out.println("SQLException : " + sqle);
            }
        }
}
```

▶ 전체 소스코드.

※ 전체 소스코드와 설명을 같이 제시하면 가독성이 좋지 않으므로, 세부 소스코드 제시와 함께 설명을 제시하였습니다.

< 세부 소스 코드 1. MS SQL Server 연동 및 Scanner 객체 생성 부분 >

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                     + "database=largeDB;integratedSecurity=true";
Connection conn = DriverManager.getConnection(connectionUrl);
Statement stmt = conn.createStatement();
System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
Scanner sc = new Scanner(System.in);
```

- ▶ database=largeDB; 를 통해, largeDB를 사용하였음을 알 수 있음.
- ▶ integratedSecurity=true; 를 통해, connectionUrl이 계속해서 사용될 수 있음을 알 수 있음.
- ▶ Statement 객체 stmt는 Connection 객체 conn의 createStatement() 메소드 호출을 통해 생성됨.
- ▶ Scanner 객체 sc를 생성하여 사용자로부터 입력을 받을 수 있도록 하였음.

< 세부 소스 코드 2. 질의 실행 및 결과 출력 부분 >

```
System.out.print("교수의 ID를 입력하세요 : ");
String i_ID = sc.nextLine();
ResultSet rs = stmt.executeQuery("SELECT distinct title FROM teaches, course "
                               + "WHERE teaches.ID = " + i_ID
                               + "and teaches.course_id = course.course_id");

System.out.println("ID가 " + i_ID + "인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...");

if (rs.next()) {
    System.out.println("해당 교수가 맡아서 진행한 강좌의 교과목명은 다음과 같습니다.");
    do {
        String field1 = rs.getString("title");
        System.out.println(field1);
    } while(rs.next());
}
else {
    System.out.println("해당 교수는 맡은 강좌가 없음.");
}
```

- ▶ sc.nextLine(); 을 통해 사용자로부터 교수의 ID를 입력 받음. 입력 받은 문자열은 교수의 ID인 i_ID 변수에 저장됨.
- ▶ ResultSet 클래스는 SQL 질의에 의해 생성된 테이블을 담는 클래스로, 객체 rs에는 stmt.executeQuery() 메소드의 인수로 전달된 질의에 의해 생성된 테이블을 담게 됨.
- ▶ 문제에서 요구한 결과를 얻기 위해 SQL 질의를 다음과 같이 작성하였으며, 추가로 WHERE절에 teaches.ID = i_ID 조건을 소스코드에 추가하여, 사용자가 입력한 교수의 ID와 같은 ID를 가진 교수에 대한 검색을 하도록 함.
- ▶ 또한, 교과목 명이 동일한 경우가 존재하므로, distinct 를 select 절에 추가하여 동일한 교과목명은 출력하지 않도록 하였음.
- ▶ rs.next()는 다음 값이 있을 경우 true, 없을 경우 false를 반환하는 메소드.
- ▶ if else 를 통해 teaches 테이블에 사용자가 입력한 ID를 가진 교수가 있을 경우와, 없을 경우에 대한 동작으로 나뉘게 됨.

※ if 문 이후 설명은 다음 페이지에 이어집니다.

▶ 있을 경우, getString() 메소드를 통해 title 속성의 값을 field1에 저장하고, 출력함. 해당 교수가 맡은 강좌를 한 행 씩 출력하게됨. while문 사용 시 처음 값이 생략되어, do while문을 사용하여 처음 값이 생략되지 않도록 함.

▶ 없을 경우, "해당 교수는 맡은 강좌가 없음." 을 출력함.

< 세부 소스 코드 4. 객체 반환 및 예외 처리 부분 >

```
sc.close();
rs.close();
stmt.close();
conn.close();
} catch (ClassNotFoundException sqle) {
System.out.println("SQLException : " + sqle);
}
```

▶ 불필요한 자원의 낭비가 없도록 close() 메소드 호출을 통해 각 객체들을 반환함.

▶ catch 블록은 SQL 예외 발생 시, 어떤 것이 발생하였는지를 출력하는 동작을 함.

< SQL문 실행 후 교수 ID 확인 >

```
select distinct ID, title
from teaches, course
where teaches.course_id = course.course_id
```

	ID	title
1	14365	Environmental Law
2	14365	The Music of the Ramones
3	15347	Systems Software
4	19368	Calculus
5	19368	International Practicum
6	19368	Shakespeare
7	22591	Animal Behavior
8	22591	Compiler Design
9	22591	FOCAL Programming
10	22591	Geology
11	22591	Graph Theory
12	22591	Greek Tragedy
13	22591	Mechanics
14	22591	The IBM 360 Architecture
15	22591	Transaction Processing
16	22591	Video Gaming
17	22591	Visual BASIC
18	25946	Drama
19	28097	Computational Biology
20	28097	Organic Chemistry
21	28400	Care and Feeding of Cats
22	28400	UNIX System Programm...

▶ 교수 ID와 과목명 확인을 위해 SQL 질의문을 실행하여 결과 테이블을 확인하였음.

< 실행 결과 – 사용자가 입력한 ID의 교수가 진행한 강좌가 존재하는 경우 >

※ 정상적인 출력 결과인지를 확인하기 위해 SQL 질의 실행 결과와 같이 제시하였습니다.

(1) ID 14365 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 ID를 입력하세요 : 14365
ID가 14365인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...
해당 교수가 맡아서 진행한 강좌의 교과목명은 다음과 같습니다.
The Music of the Ramones
Environmental Law
```

	ID	title
1	14365	The Music of the Ramones
2	14365	Environmental Law

(2) ID 22591 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 ID를 입력하세요 : 22591
ID가 22591인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...
해당 교수가 맡아서 진행한 강좌의 교과목명은 다음과 같습니다.
Animal Behavior
Compiler Design
FOCAL Programming
Geology
Graph Theory
Greek Tragedy
Mechanics
The IBM 360 Architecture
Transaction Processing
Video Gaming
Visual BASIC
```

	title
1	Animal Behavior
2	Compiler Design
3	FOCAL Programming
4	Geology
5	Graph Theory
6	Greek Tragedy
7	Mechanics
8	The IBM 360 Architecture
9	Transaction Processing
10	Video Gaming
11	Visual BASIC

▶ 두 경우 모두 정상적으로 입력된 ID를 가진 교수가 진행한 강좌명을 출력하였으며, MS SQL Server에서 실행한 결과와 JAVA 프로그램에서 실행한 결과가 같음을 확인하였음.

	title
1	Graph Theory
2	Graph Theory
3	Compiler Design
4	Visual BASIC
5	Visual BASIC
6	FOCAL Progra...
7	Mechanics
8	Video Gaming
9	Geology
10	The IBM 360 A...
11	Animal Behavior
12	Greek Tragedy
13	Transaction Pr...

▶ 추가로, distinct를 사용하지 않았을 때의 결과를 확인하였음.

▶ 중복된 결과를 제외하고 정상적으로 출력하였음을 확인하였음.

< 실행 결과 – 사용자가 입력한 ID의 교수가 진행한 강좌가 존재하지 않는 경우 >

※ 정상적인 출력 결과인지를 확인하기 위해 SQL 질의 실행 결과와 같이 제시하였습니다.

(3) ID 16453 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 ID를 입력하세요 : 16453
ID가 16453인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...
해당 교수는 맡은 강좌가 없음.
```

title

▶ 16453이라는 ID 값을 가진 교수가 위 결과 릴레이션에 없기 때문에, "해당 교수는 맡은 강좌가 없음." 이라는 메세지를 출력하고 프로그램이 종료되었음을 확인하였음.

2. VIEW dept_section_cnt 를 정의하시오. dept_section_cnt 는 학과+학년도 별로 지금까지 개설한 강좌수를 조회할 수 있도록 정의하시오. (← VIEW 정의는 sqlserver 에서 직접 하시오.) 학과명을 사용자로부터 입력받아 VIEW dept_section_cnt 로부터 해당 학과가 지금까지 개설한 강좌수를 학년도별로 조회하는 JAVA 프로그램을 작성하시오. 해당 학과가 지금까지 개설한 강좌가 하나도 없을 경우 이러한 사실도 알려줄 수 있도록 하시오.

< VIEW dept_section_cnt 생성 >

```
create view dept_section_cnt as
select dept_name, year, count(*) as cnt_courses
from section, course
where section.course_id = course.course_id
group by dept_name, year
```

▶ section 테이블과 course 테이블을 이용해 dept_section_cnt 뷰를 생성하였음.

```
select *
from dept_section_cnt
```

	dept_name	year	cnt_courses
1	Biology	2001	1
2	Cybernetics	2001	1
3	Finance	2001	1
4	History	2001	2
5	Accounting	2002	1
6	Astronomy	2002	1
7	Biology	2002	1
8	Civil Eng.	2002	1
9	Comp. Sci.	2002	1
10	Cybernetics	2002	1
11	Elec. Eng.	2002	1
12	Finance	2002	1
13	Languages	2002	1
14	Marketing	2002	1
15	Math	2002	2
16	Physics	2002	1
17	Accounting	2003	2
18	Athletics	2003	1

▶ 뷰를 통해 학과, 학년도 별 강좌 수 조회를 할 수 있음을 확인하였음.

< 전체 소스 코드 >

```
public static void main(String[] args) throws ClassNotFoundException, SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                                 + "database=largeDB;"
                                 + "integratedSecurity=true";
            Connection conn = DriverManager.getConnection(connectionUrl);
            Statement stmt = conn.createStatement();
            System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
            Scanner sc = new Scanner(System.in);

            System.out.print("학과 이름을 입력하세요 : ");
            String dept_name = sc.nextLine();
            ResultSet rs = stmt.executeQuery("SELECT * "
                                           + "FROM dept_section_cnt "
                                           + "WHERE dept_name = '" + dept_name + "'");

            System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의 수를 계산 중 입니다...");

            if (rs.next()) {
                System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의 수는 다음과 같습니다.");
                System.out.println("학과명\t학년도\t강좌 수");
                do {
                    String field1 = rs.getString("dept_name");
                    String field2 = rs.getString("year");
                    String field3 = rs.getString("cnt_courses");
                    System.out.print(field1 + "\t");
                    System.out.print(field2 + "\t");
                    System.out.println(field3);
                } while(rs.next());
            }
            else {
                System.out.println("해당 학과는 지금까지 개설한 강좌가 없습니다!");
            }

            sc.close();
            rs.close();
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException sqle) {
            System.out.println("SQLException : " + sqle);
        }
```

▶ 전체 소스코드.

※ 전체 소스코드와 설명을 같이 제시하면 가독성이 좋지 않으므로, 세부 소스코드 제시와 함께 설명을 제시하였습니다.

< 세부 소스 코드 1. MS SQL Server 연동 및 Scanner 객체 생성 부분 >

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                     + "database=largeDB;integratedSecurity=true";
Connection conn = DriverManager.getConnection(connectionUrl);
Statement stmt = conn.createStatement();
System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
Scanner sc = new Scanner(System.in);
```

▶ database=largeDB; 를 통해, largeDB를 사용하였음을 알 수 있음.

▶ integratedSecurity=true; 를 통해, connectionUrl이 계속해서 사용될 수 있음을 알 수 있음.

▶ Statement 객체 stmt는 Connection 객체 conn의 createStatement() 메소드 호출을 통해 생성됨.

▶ Scanner 객체 sc를 생성하여 사용자로부터 입력을 받을 수 있도록 하였음.

< 세부 소스 코드 2. 질의 실행 및 결과 출력 부분 >

```
System.out.print("학과 이름을 입력하세요 : ");
String dept_name = sc.nextLine();
ResultSet rs = stmt.executeQuery("SELECT * "
                               + "FROM dept_section_cnt "
                               + "WHERE dept_name = '" + dept_name + "'");

System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의 수를 계산 중 입니다...");

if (rs.next()) {
    System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의 수는 다음과 같습니다.");
    System.out.println("학과명\t학년도\t강좌 수");
    do {
        String field1 = rs.getString("dept_name");
        String field2 = rs.getString("year");
        String field3 = rs.getString("cnt_courses");
        System.out.print(field1 + "\t");
        System.out.print(field2 + "\t");
        System.out.println(field3);
    } while(rs.next());
}
else {
    System.out.println("해당 학과는 지금까지 개설한 강좌가 없습니다!");
}
```

▶ sc.nextLine(); 을 통해 사용자로부터 학과 이름을 입력 받음. 입력 받은 문자열은 학과 이름인 dept_name에 저장되게 됨.

▶ ResultSet 클래스는 SQL 질의에 의해 생성된 테이블을 담는 클래스로, 객체 rs에는 stmt.executeQuery() 메소드의 인수로 전달된 질의에 의해 생성된 테이블을 담게 됨.

▶ 문제에서 요구한 결과를 얻기 위해 SQL 질의를 다음과 같이 작성하였으며, 추가로 WHERE절에 dept_name = dept_name 조건을 소스코드에 추가하여, 사용자가 입력한 학과 이름에 대한 검색을 하도록 함.

▶ rs.next()는 다음 값이 있을 경우 true, 없을 경우 false를 반환하는 메소드.

▶ if else 를 통해 dept_section_cnt 뷰에 사용자가 입력한 학과 이름이 있을 경우와, 없을 경우에 대한 동작으로 나뉘게 됨.

※ if 문 이후 설명은 다음 페이지에 이어집니다.

▶ 있을 경우, getString() 메소드를 통해 dept_name 속성의 값을 field1에 저장. year 속성의 값을 field2에 저장, cnt_courses 속성의 값을 field3에 저장하고, 한 행에 세 속성의 값이 함께 출력됨. while문 사용 시 처음 값이 생략되어, do while문을 사용하여 처음 값이 생략되지 않도록 함.

▶ 없을 경우, "해당 학과는 지금까지 개설한 강좌가 없습니다!" 를 출력함.

< 세부 소스 코드 3. 객체 반환 및 예외 처리 부분 >

```
sc.close();
rs.close();
stmt.close();
conn.close();
} catch (ClassNotFoundException sqle) {
System.out.println("SQLException : " + sqle);
}
```

▶ 불필요한 자원의 낭비가 없도록 close() 메소드 호출을 통해 각 객체들을 반환함.

▶ catch 블록은 SQL 예외 발생 시, 어떤 것이 발생하였는지를 출력하는 동작을 함.

< 실행 결과 – 사용자가 입력한 학과가 개설한 강좌가 존재하는 경우 >

※ 정상적인 출력 결과인지를 확인하기 위해 SQL 질의 실행 결과와 같이 제시하였습니다.

(1) 'Comp. Sci.' 학과 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
학과 이름을 입력하세요 : Comp. Sci.
Comp. Sci. 학과가 학년도 별로 지금까지 개설한 강좌의 수를 계산 중 입니다...
Comp. Sci. 학과가 학년도 별로 지금까지 개설한 강좌의 수는 다음과 같습니다.
학과명    학년도    강좌 수
Comp. Sci.      2002    1
Comp. Sci.      2004    2
Comp. Sci.      2007    1
```

	dept_name	year	cnt_courses
1	Comp. Sci.	2002	1
2	Comp. Sci.	2004	2
3	Comp. Sci.	2007	1

(2) 'Finance' 학과 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
학과 이름을 입력하세요 : Finance
Finance 학과가 학년도 별로 지금까지 개설한 강좌의 수를 계산 중 입니다...
Finance 학과가 학년도 별로 지금까지 개설한 강좌의 수는 다음과 같습니다.
학과명    학년도    강좌 수
Finance 2001    1
Finance 2002    1
Finance 2003    1
Finance 2004    1
Finance 2005    1
Finance 2006    2
Finance 2008    2
Finance 2009    1
```

	dept_name	year	cnt_courses
1	Finance	2001	1
2	Finance	2002	1
3	Finance	2003	1
4	Finance	2004	1
5	Finance	2005	1
6	Finance	2006	2
7	Finance	2008	2
8	Finance	2009	1

▶ 두 경우 모두 정상적으로 학과명과 해당 년도, 강좌 수를 출력함을 확인하였으며, MS SQL Server에서 실행한 결과와 JAVA 프로그램에서 실행한 결과가 동일함을 확인하였음.

< 실행 결과 – 사용자가 입력한 학과가 개설한 강좌가 존재하지 않는 경우 >

※ 정상적인 출력 결과인지를 확인하기 위해 SQL 질의 실행 결과와 같이 제시하였습니다.

(3) 'Mech. Eng.' 학과 입력 시

```
select distinct dept_name
from dept_section_cnt
```

	dept_name
1	Accounting
2	Astronomy
3	Athletics
4	Biology
5	Civil Eng.
6	Comp. Sci.
7	Cybernetics
8	Elec. Eng.
9	English
10	Finance
11	Geology
12	History
13	Languages
14	Marketing
15	Math
16	Physics
17	Pol. Sci.
18	Psychology
19	Statistics

```
select distinct dept_name
from course
```

	dept_name
1	Accounting
2	Astronomy
3	Athletics
4	Biology
5	Civil Eng.
6	Comp. Sci.
7	Cybernetics
8	Elec. Eng.
9	English
10	Finance
11	Geology
12	History
13	Languages
14	Marketing
15	Math
16	Mech. Eng.
17	Physics
18	Pol. Sci.
19	Psychology
20	Statistics

▶ 왼쪽은 현재까지 개설한 강좌가 있는 학과의 결과이며, 오른쪽은 LargeDB에 존재하는 학과의 결과로, 'Mech. Eng.' 학과가 현재까지 개설한 강좌가 없음을 확인하였음.

```
MS-SQL 서버 접속에 성공하였습니다.
학과 이름을 입력하세요 : Mech. Eng.
Mech. Eng. 학과가 학년도 별로 지금까지 개설한 강좌의 수를 계산 중 입니다...
해당 학과는 지금까지 개설한 강좌가 없습니다!
```

dept_name

▶ Mech. Eng. 학과가 현재까지 개설한 강좌가 없으므로, "해당 학과는 지금까지 개설한 강좌가 없습니다!" 라는 메세지가 정상적으로 출력되었음을 확인하였음.

4. 함수 inst_advise_student(@inst_name varchar(20))를 작성하여 largeDB 의 함수자원으로 저장하시오. 함수 inst_advise_student 는 교수이름을 @inst_name 으로 전달받은 후 해당 교수가 지도하는 학생 수를 return 하는 함수이다. 단, 교수는 동명이인이 존재하지 않는 것으로 한다. 사용자로부터 교수이름을 입력받은 후 largeDB 에 저장되어 있는 inst_advise_student 를 호출하여 해당 교수가 지도하는 학생 수를 알려주는 JAVA 프로그램을 작성하시오.

< 함수 inst_advise_student(@inst_name varchar(20)) 작성 >

```
create function inst_advise_student(@inst_name varchar(20))
returns integer
as
begin
declare @s_count integer
select @s_count=count(*)
from advisor as A, instructor as I
where A.i_ID = I.ID and I.name = @inst_name
return @s_count
end
```

▶ advisor, instructor 테이블에서 교수의 ID가 같고, inst_name 과 instructor 테이블의 name 이 같은 튜플의 수(= 학생의 수)를 count하여 s_count로 반환함.

< 전체 소스 코드 >

```
public class report_03 {
    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                                 + "database=largeDB;integratedSecurity=true";
            Connection conn = DriverManager.getConnection(connectionUrl);
            System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
            Scanner sc = new Scanner(System.in);

            System.out.println("교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그램입니다.");
            System.out.print("교수의 이름을 입력하세요 : ");
            String inst_name = sc.nextLine();

            CallableStatement cstmt = conn.prepareCall("{? = call dbo.inst_advise_student (?) }");
            cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
            cstmt.setString(2, inst_name);
            cstmt.execute();

            String output = cstmt.getString(1);
            System.out.println(inst_name + " 교수가 지도하는 총 학생 수는 " + output + " 명 입니다.");

            sc.close();
            cstmt.close();
            conn.close();
        } catch (ClassNotFoundException sqle) {
            System.out.println("SQLException : " + sqle);
        }
    }
}
```

▶ 전체 소스코드.

※ 전체 소스코드와 설명을 같이 제시하면 가독성이 좋지 않으므로, 세부 소스코드 제시와 함께 설명을 제시하였습니다.

< 세부 소스 코드 1. MS SQL Server 연동 및 Scanner 객체 생성 부분 >

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                     + "database=largeDB;integratedSecurity=true";
Connection conn = DriverManager.getConnection(connectionUrl);
System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
Scanner sc = new Scanner(System.in);
```

- database=largeDB; 를 통해, largeDB를 사용하였음을 알 수 있음.
- integratedSecurity=true; 를 통해, connectionUrl이 계속해서 사용될 수 있음을 알 수 있음.
- Scanner 객체 sc를 생성하여 사용자로부터 입력을 받을 수 있도록 하였음.

< 세부 소스 코드 2. 질의 실행 및 결과 출력 부분 >

```
System.out.println("교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그램입니다.");
System.out.print("교수의 이름을 입력하세요 : ");
String inst_name = sc.nextLine();

CallableStatement cstmt = conn.prepareCall("{? = call dbo.inst_advise_student (?) }");
cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
cstmt.setString(2, inst_name);
cstmt.execute();

String output = cstmt.getString(1);
System.out.println(inst_name + " 교수가 지도하는 총 학생 수는 " + output + " 명 입니다.");
```

- sc.nextLine(); 을 통해 사용자로부터 교수 이름을 입력 받음. 입력 받은 문자열은 교수 이름인 inst_name에 저장되게 됨.
- 함수 사용을 위해 CallableStatement 객체 cstmt를 생성하고, Connection 객체 conn의 prepareCall() 메소드에 inst_adivse_student(@inst_name) 함수를 인수로 전달하였음. 이로써 JAVA 프로그램에서 LargeDB의 함수 inst_advise_student(@inst_name) 호출이 가능함.
- 문제에서 요구한 결과를 얻기 위해 출력 파라미터를 1번, 정수타입으로 설정하고, setString() 메소드에 2번, inst_name으로 설정하고, execute() 메소드를 통해 함수를 실행하였음.
- output 문자열에는 getString() 메소드를 통해 파라미터 1번의 값. 즉 입력한 교수에게 지도를 받는 학생들의 수가 저장됨.

< 세부 소스 코드 3. 객체 반환 및 예외 처리 부분 >

```
    sc.close();
    cstmt.close();
    conn.close();
} catch (ClassNotFoundException sqle) {
    System.out.println("SQLException : " + sqle);
}
```

- 불필요한 자원의 낭비가 없도록 close() 메소드 호출을 통해 각 객체들을 반환함.
- catch 블록은 SQL 예외 발생 시, 어떤 것이 발생하였는지를 출력하는 동작을 함.

< 실행 결과 >

※ 정상적인 출력 결과인지를 확인하기 위해 SQL 질의 실행 결과와 같이 제시하였습니다.

(1) 'Lembr' 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그램입니다.
교수의 이름을 입력하세요 : Lembr
Lembr 교수가 지도하는 총 학생 수는 39 명 입니다.
```

	s_count
1	39

(2) 'DAgostino' 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그램입니다.
교수의 이름을 입력하세요 : DAgostino
DAgostino 교수가 지도하는 총 학생 수는 40 명 입니다.
```

	s_count
1	40

▶ Lembr 교수가 지도하는 학생의 수는 총 39명, DAgostino 교수가 지도하는 학생의 수는 총 40명인 것을 확인하였음. SQL Server에서의 실행 결과도 동일함을 확인하였음.

(3) 'YoungHwan' 입력 시

```
MS-SQL 서버 접속에 성공하였습니다.
교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그램입니다.
교수의 이름을 입력하세요 : YoungHwan
YoungHwan 교수가 지도하는 총 학생 수는 0 명 입니다.
```

	s_count
1	0

▶ YougHwan 이라는 교수는 사실 존재하지 않는 교수이나, 그러한 경우에 대한 처리는 없기 때문에 0을 반환해주는 것을 확인하였음.

< 소스 코드 >

1번 소스코드

```
import java.sql.*;
import java.sql.SQLException;
import java.util.Scanner;

public class report_01 {
	public static void main(String[] args) throws ClassNotFoundException, SQLException {
		try {
		Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
		String connectionUrl = "jdbc:sqlserver://localhost:14160;"
				+ "database=largeDB;integratedSecurity=true";
		Connection conn = DriverManager.getConnection(connectionUrl);
		Statement stmt = conn.createStatement();
		System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
		Scanner sc = new Scanner(System.in);

		System.out.print("교수의 ID를 입력하세요 : ");
		String i_ID = sc.nextLine();
		ResultSet rs = stmt.executeQuery("SELECT distinct title FROM teaches, course "
				+"WHERE teaches.ID = " + i_ID
				+ "and teaches.course_id = course.course_id");

		System.out.println("ID가 " + i_ID + "인 교수가 맡아서 진행한 강좌의 교과목명을 검색합니다...");

		if (rs.next()) {
			System.out.println("해당 교수가 맡아서 진행한 강좌의 교과목명은 다음과 같습니다.");
			do {
				String field1 = rs.getString("title");
				System.out.println(field1);
			} while(rs.next());
		}
		else {
			System.out.println("해당 교수는 맡은 강좌가 없음.");
		}

		sc.close();
		rs.close();
		stmt.close();
		conn.close();
		} catch (ClassNotFoundException sqle) {
	System.out.println("SQLException : " + sqle);
	}
	}
}
```

2,3 번 소스코드

```
import java.sql.*;
import java.util.Scanner;

public class report_02 {

	public static void main(String[] args) throws ClassNotFoundException, SQLException {
		try {
			Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
			String connectionUrl = "jdbc:sqlserver://localhost:14160;"
					+ "database=largeDB;"
					+ "integratedSecurity=true";
			Connection conn = DriverManager.getConnection(connectionUrl);
			Statement stmt = conn.createStatement();
			System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
			Scanner sc = new Scanner(System.in);

			System.out.print("학과 이름을 입력하세요 : ");
			String dept_name = sc.nextLine();
			ResultSet rs = stmt.executeQuery("SELECT * "
					+ "FROM dept_section_cnt "
					+ "WHERE dept_name = '" + dept_name + "'");

			 System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의 수를
계산 중 입니다...");

			if (rs.next()) {
				System.out.println(dept_name + " 학과가 학년도 별로 지금까지 개설한 강좌의
수는 다음과 같습니다.");
				System.out.println("학과명₩t학년도₩t강좌 수");
				do {
					String field1 = rs.getString("dept_name");
					String field2 = rs.getString("year");
					String field3 = rs.getString("cnt_courses");
					System.out.print(field1 + "₩t");
					System.out.print(field2 + "₩t");
					System.out.println(field3);
				} while(rs.next());
			}
			else {
				System.out.println("해당 학과는 지금까지 개설한 강좌가 없습니다!");
			}

			sc.close();
			rs.close();
```

```
                stmt.close();
                conn.close();
            } catch (ClassNotFoundException sqle) {
              System.out.println("SQLException : " + sqle);
            }
        }

}
```

4번 소스코드

```
import java.sql.*;
import java.util.Scanner;

public class report_03 {
    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            String connectionUrl = "jdbc:sqlserver://localhost:14160;"
                                    + "database=largeDB;integratedSecurity=true";
            Connection conn = DriverManager.getConnection(connectionUrl);
            System.out.println("MS-SQL 서버 접속에 성공하였습니다.");
            Scanner sc = new Scanner(System.in);

            System.out.println("교수의 이름을 입력하면 해당 교수가 지도하는 학생의 수를 출력하는 프로그
램입니다.");
            System.out.print("교수의 이름을 입력하세요 : ");
            String inst_name = sc.nextLine();

            CallableStatement cstmt = conn.prepareCall("{? = call dbo.inst_advise_student (?) }");
            cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
            cstmt.setString(2, inst_name);
            cstmt.execute();

            String output = cstmt.getString(1);
            System.out.println(inst_name + " 교수가 지도하는 총 학생 수는 " + output + " 명 입니다.");

            sc.close();
            cstmt.close();
            conn.close();
        } catch (ClassNotFoundException sqle) {
          System.out.println("SQLException : " + sqle);
        }
    }
}
```