7️⃣

Team07 인간 DB

팀원 🙋‍♀️

2071010 김선영

2076033 김도연

2076071 김용연

2076292 이나현

2071044 정민정

목차

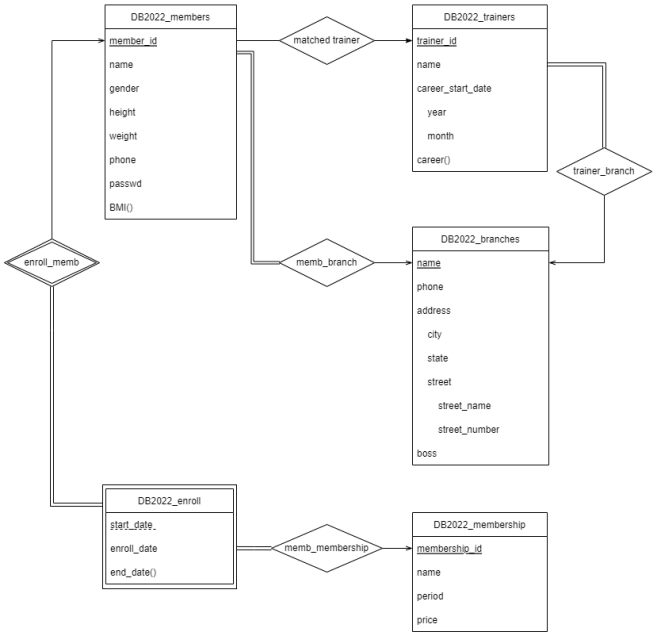
1. 요구 분석

> 헬스장 데이터베이스 필요성 설명 및 요구 분석

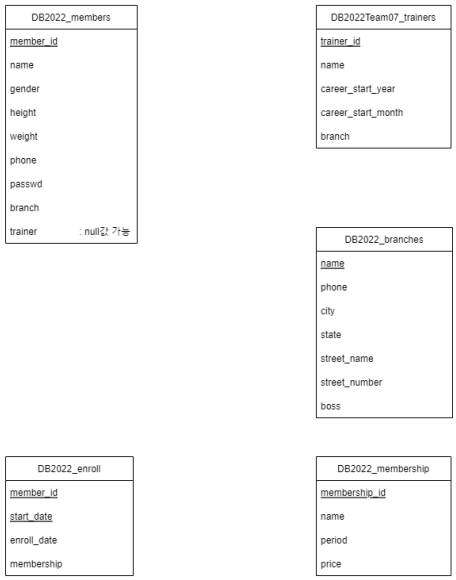
- 헬스장에는 지점 데이터, 활동회원과 휴식회원 데이터, 근무하는 트레이너 데이터, 회원권 데이터 등 다양한 데이터가 서로 복잡한 관계를 맺으며 존재한다.
- 다양한 관계를 보이는 데이터를 통합하여 관리하는 데이터베이스가 필요하다고 생각되어 헬스장을 적합한 기관으로 선택하였다.
- 또한 최근에는 코로나 규제가 점점 완화되며 헬스장을 찾는 사람들이 급격히 증가했으므로, 데이터 관리 프로그램의 사용이 필수적으로 보인다.

2. ER Diagram과 Relation Schema

[ER Diagram]



[Relation Schema]



5️⃣ Entity set은 총 5개이다.

- DB2022_members는 휴식 회원과 활동 회원을 포함한 모든 회원에 대한 정보,
- DB2022_enroll은 활동 회원에 대한 정보,
- DB2022_trainers는 트레이너에 대한 정보,
- DB2022_branches는 헬스장의 지점에 대한 정보,
- DB2022_membership은 회원권에 대한 정보를 지니고 있다.

2-1. Derived Attribute 설명

- **DB2022_enroll의 end_date()**
  : 회원권이 끝나는 날짜인 end_date는 start_date와 사용중인 회원권의 기간(DB2022_membership의 period)을 이용하여 구할 수 있다.

- **DB2022_members의 BMI()**
  : 회원의 BMI 점수인 BMI는 height와 weight를 이용하여 구할 수 있다.

- **DB2022_trainers의 career()**
  : 트레이너의 경력인 career는 트레이너가 일을 시작한 날짜인 career_start_date와 현재 날짜를 이용하여 구할 수 있다. career는 월 단위로 표현된다.

2-2. Relationship 설명

- **enroll_memb**

: DB2022_enroll은 활동 회원에 대한 entity set이므로, 모든 회원에 대한 entity set인 DB2022_members와 정보가 중복된다. 따라서 DB2022_enroll은 DB2022_members에 의존하는 weak entity set이다. 한 회원은 하루에 2개 이상의 회원권을 사용할 수 없으므로, 회원권을 사용하기 시작한 날짜인 start_date는 discriminator가 된다.

- **matched trainer**

  : 활동 회원은 누구나 트레이너에게 PT를 받을 수 있다. 한 명의 트레이너는 여러 명의 회원을 맡을 수 있으며, 한 명의 회원은 한 명의 트레이너에게만 PT를 받을 수 있다. 따라서 DB2022_trainers와 DB2022_memberss는 One-to-Many Relationship이다. 휴식 회원이거나 PT를 받지 않는 활동 회원은 담당 트레이너가 존재하지 않으므로, Partial Participation이다.

- **memb_branch**

  : 한 명의 회원은 반드시 하나의 지점에 소속된다. 하나의 지점엔 여러 명의 회원이 소속될 수 있으므로, DB2022_branches와 DB2022_members는 One-to-Many Relationship이다. 지점에 소속되지 않은 회원은 존재하지 않으므로, Total Participation이다.

- **trainer_branch**

  : 한 명의 트레이너는 반드시 하나의 지점에 소속된다. 하나의 지점엔 여러 명의 트레이너가 소속될 수 있으므로, DB2022_branches와 DB2022_trainers는 One-to-Many Relationship이다. 지점에 소속되지 않은 트레이너는 존재하지 않으므로, Total Participation이다.

- **memb_membership**

  : 한 명의 활동 회원은 반드시 하나의 회원권을 가진다. 하나의 회원권은 여러 명의 회원이 살 수 있으므로, DB2022_membership과 DB2022_enroll은 One-to-Many Relationship이다. 회원권을 구매하지 않은 활동 회원은 존재하지 않으므로, Total Participation이다.

2-3. Relation Schema

1. enroll_memb

DB2022_enroll은 DB2022_members를 identify entity set으로 하는 weak entity set이다. 따라서 DB2022_enroll의 discriminator인 start_date와 DB2022_members의 primary key인 member_id를 DB2022_enroll의 primary key로 설정했다.

2. matched_trainer

DB2022_trainers와 DB2022_members는 One-to-Many Relationship이고, Partial Participation이므로, DB2022_members에 DB2022_trainers의 primary key를 null값이 가능한 attribute로 추가했다.

3. memb_branch

DB2022_branches와 DB2022_members는 One-to-Many Relationship이고, Total Participation이므로, DB2022_branches의 primary key를 DB2022_members의 attribute로 추가했다.

4. trainer_branch

DB2022_branches와 DB2022_trainers는 One-to-Many Relationship이고, Total Participation이므로, DB2022_branches의 primary key를 DB2022_trainers의 attribute로 추가했다.

5. memb_membership

DB2022_branches와 DB2022_trainers는 One-to-Many Relationship이고, Total Participation이므로, DB2022_branches의 primary key를 DB2022_members의 attribute로 추가했다.

3. Java 코드 설명

> 자바 코드의 클래스(class)와 생성자 및 메서드(method) 설명

Class DB2022Team07_main

: MainFrame 객체를 생성하는 main 함수가 있는 클래스이다.

Class MainFrame

: 메인 화면 GUI를 보여주는 클래스이다.

- **MainFrame**()
  - 총 6개의 서비스를 제공한다
  - Branch Information(지점 정보 조회), Sign Up(회원 가입), Buy Membership(회원권 구매), My Page(회원 페이지), PT Application / Change Trainer(PT 신청/트레이너 변경), Staff Page(직원 페이지)).
  - 각 서비스 메뉴를 선택하면, 해당 서비스를 제공하는 페이지를 띄운다.

Class RegisterFrame

: 회원 가입 화면 GUI를 보여주는 클래스이다.

- **RegisterFrame**()
  - 입력 정보가 유효한지 판단하고, 회원 가입을 진행한다.
- **Register**(**String** name, **String** phone, **String** passwd, **String** gender, **String** height, **String** weight, **String** branch)
  - 필수로 입력되어야 하는 사용자 정보를 받아 DB2022_members에 추가하여 회원 가입을 완료한다.
  - trainer는 null로 생성한다.

Class MyPageFrame

: 회원 페이지 GUI를 보여주는 클래스이다.

- **MyPageFrame**()
  - 총 3개의 서비스를 제공한다
  - BMI Calculation(BMI 계산), Membership Expiration Date Inquiry(회원권 만료 날짜 조회), Edit Member Information(회원 정보 수정)
  - Edit Member Information 메뉴에서는 password, height, weight 정보를 변경할 수 있다.

Class BranchFrame

: 지점별 정보 조회 페이지 GUI를 보여주는 클래스이다.

- **BranchFrame**()
  - 총 4개의 서비스를 제공한다
  - Monthly Sales per Branch(지점 별 월 매출 조회), Number of Members per Branch(지점 별 회원 수 조회), Number of Trainers per Branch(지점 별 트레이너 수 조회), Top 1 Trainer per Branch(지점 별 회원수 1위 트레이너 조회).
  - 지점 별 월 매출 조회 메뉴에서는 조회하고자 하는 연도와 월을 입력하면, 선택한 월의 매출을 보여준다.
- **Monthly_Revenue**()
  - 지점 별로 입력한 날짜에 해당하는 매출을 계산한다.
  - 활동 회원들의 회원권 등록 날짜(DB2022_enroll의 enroll_date)를 기준으로, 회원권 가격을 합산하여 구한다.

Class TrainerFrame

: 트레이너 등록/변경 페이지 GUI를 보여주는 클래스이다.

- **TrainerFrame**()
  - 트레이너를 새로 등록하거나 기존 트레이너를 변경하는 서비스를 진행한다.
  - PT는 활동 회원에게만 제공되는 서비스이므로, 활동 회원이 맞는지 본인 확인을 진행한 후에 서비스를 제공한다.
- **Trainer_Change**(**String** member_name, **String** phone, **String** passwd)
  - 회원이 소속된 지점의 트레이너 목록을 보여준 다음, 새롭게 등록하고자 하는 트레이너의 번호를 입력 받는다.
  - 목록에 있는 트레이너의 번호를 입력해야만 트레이너가 등록 혹은 변경된다.

Class EmployeeFrame

: 직원 페이지 GUI를 보여주는 클래스이다.

- **EmployeeFrame**()
  - 크게 트레이너를 위한 서비스와 관리자를 위한 서비스를 제공한다.
  - 트레이너를 위한 서비스는 총 3가지를 제공한다: Resignation(트레이너 퇴사 신청), Transfer Branch(트레이너 지점 이동 신청), Change Password(비밀번호 변경).
  - 관리자를 위한 서비스는 총 1가지를 제공한다: Delete Expired Member(회원권이 만료된 회원을 휴식회원으로 전환).
- **Trainer_Quit**(**String** trainer)
  - 퇴사한 트레이너가 담당한 회원의 DB2022_members.trainer를 null로 변경한 후, DB2022_trainers에서 트레이너 정보를 삭제한다.
  - 2개의 query를 Transaction으로 설정하여 퇴사한 트레이너가 담당한 회원의 trainer field만 null로 변경되거나 트레이너 정보만 삭제되는 상황을 예방한다.
- **Trainer_Move**(**String** trainer, **String** branch)
  - 트레이너가 담당한 회원의 DB2022_members.trainer를 null로 변경한 후, 트레이너의DB2022_trainers.branch를 변경한다.
  - 2개의 query를 Transaction으로 설정하여 트레이너가 담당한 회원의 trainer field만 null로 변경되거나 트레이너의 지점만 변경되는 상황을 예방한다.
- **change_pwd**(**String** pwd)
  - 넘겨 받은 pwd 변수로 트레이너의 비밀번호를 변경한다.
  - 4자리의 문자열을 입력해야만 비밀번호가 변경된다.
- **delete_member**()
  - 전 지점의 회원을 대상으로 회원권이 만료된 회원을 DB2022_enroll에서 삭제한다.

Class EnrollFrame

: 회원권 등록 페이지 GUI를 생성하는 클래스이다.

- **EnrollFrame**()
  - 입력한 정보가 유효한지 확인하고, 회원권을 등록한다.
- **signIn**(**String** member_name, **String** phone, **String** passwd)
  - 회원에 대한 본인 인증을 진행한다.
- **enrollMembership**(**java.sql.Date** start_date, **String** membership_id)
  - 입력 받은 회원권 번호와 시작 날짜를 기준으로 DB2022_enroll에 회원권을 등록한다.
  - 등록 날짜인 DB2022_enroll.enroll_date는 현재 날짜로 입력한다.

- **showEnrollField**(**Boolean** b)
  - 회원권 등록 화면 component를 보이거나 안 보이게 한다.

4. 응용프로그램 사용 방법

> 응용프로그램 설치 및 사용 방법, 연결을 위한 구성

`mysql-connector-java.jar` 을 이용하여 mysql 과 Java 코드를 연동하여 작동되는 파일이다.

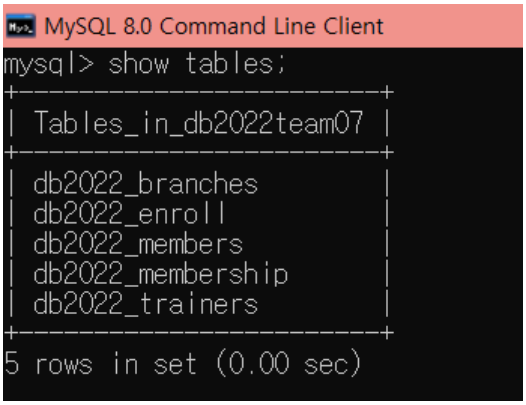
5. 17개 요구 사항 설명

> 요구 사항과 관련된 부분의 코드와 사용자 인터페이스 스크린 작동 화면

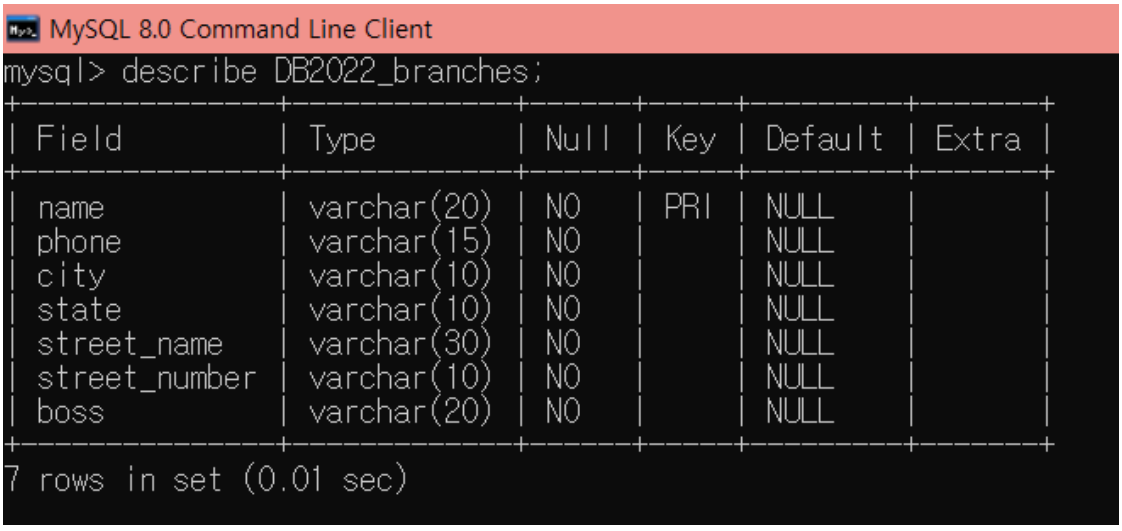
---

1. 5개 이상의 테이블을 가지고 있어야 하고 각 테이블들의 컬럼(Attribute)의 수를 합하면 20개 이상이어야 한다.

- **총 Table 수 5개**



- **총 Arrtibute 수 30개**
  - DB2022_branches : 7개



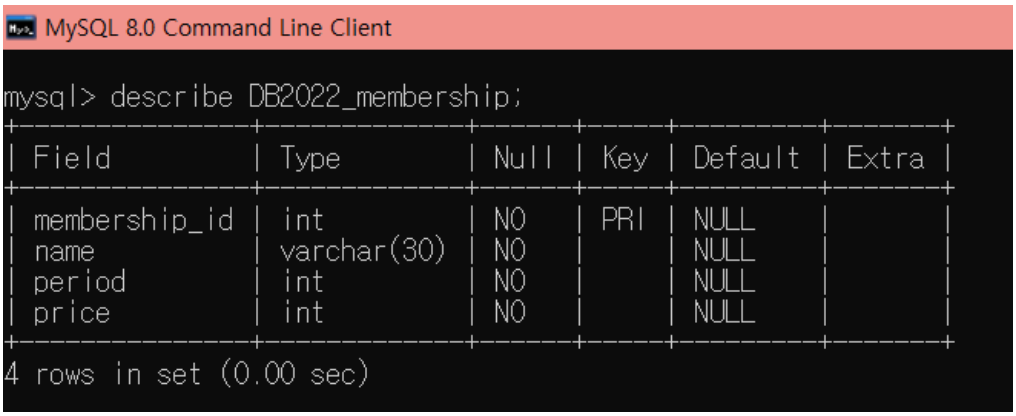
- DB2022_enroll : 4개

```
mysql> describe DB2022_enroll;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| member_id   | int  | NO   | PRI | NULL    |       |
| enroll_date | date | NO   |     | NULL    |       |
| start_date  | date | NO   | PRI | NULL    |       |
| membership  | int  | NO   | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

- DB2022_members : 9개

```
mysql> describe DB2022_members;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| member_id | int         | NO   | PRI | NULL    |       |
| name      | varchar(20) | NO   |     | NULL    |       |
| gender    | char(1)     | NO   |     | NULL    |       |
| height    | int         | NO   |     | NULL    |       |
| weight    | int         | NO   |     | NULL    |       |
| phone     | char(11)    | NO   | UNI | NULL    |       |
| password  | varchar(4)  | NO   |     | NULL    |       |
| branch    | varchar(20) | NO   | MUL | NULL    |       |
| trainer   | int         | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

- DB2022_membership : 4개

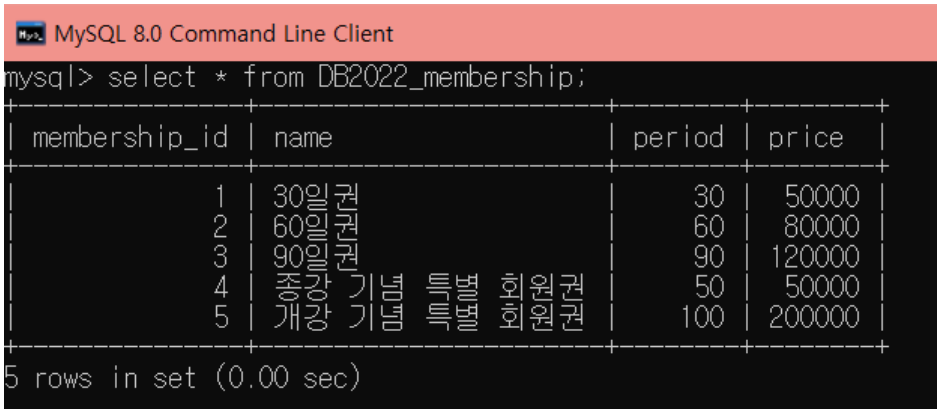


- DB2022_trainers : 6개

```
mysql> describe DB2022_trainers;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| trainer_id         | int         | NO   | PRI | NULL    |       |
| name               | varchar(20) | NO   |     | NULL    |       |
| career_start_year  | int         | NO   |     | NULL    |       |
| career_start_month | int         | NO   |     | NULL    |       |
| branch             | varchar(30) | NO   | MUL | NULL    |       |
| password           | varchar(4)  | NO   |     | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

2. 초기화를 위해 적어도 30개의 레코드(투플)를 가지고 있어야 한다. (모든 테이블들의 레코드 수의 총합이 30개 이상)

- **총 레코드 수 32개**
  - DB2022_membership : 5개



  - DB2022_branches : 3개

  - DB2022_trainers : 6개

```
mysql> select * from DB2022_trainers;
+------------+--------+-------------------+--------------------+--------+----------+
| trainer_id | name   | career_start_year | career_start_month | branch | password |
+------------+--------+-------------------+--------------------+--------+----------+
|          1 | 김진해 |              2017 |                  5 | 수지   | 30q0     |
|          2 | 정완섭 |              2022 |                  6 | 수지   | 5ab0     |
|          3 | 도실인 |              2013 |                 11 | 영창   | 3068     |
|          4 | 채준기 |              2010 |                 12 | 노량진 | Ai39     |
|          5 | 김정연 |              2020 |                  7 | 노량진 | 1c59     |
|          6 | 이은혜 |              2015 |                  9 | 영창   | 7e4r     |
+------------+--------+-------------------+--------------------+--------+----------+
6 rows in set (0.00 sec)
```

- DB2022_members : 11개

```
mysql> select * from DB2022_members;
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
| member_id | name   | gender | height | weight | phone       | password | branch | trainer |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
|         1 | 이나현 | F      |    160 |     60 | 01029302038 | 2av0     | 수지   |       1 |
|         2 | 김선영 | F      |    170 |     70 | 01034062059 | 1930     | 노량진 |       4 |
|         3 | 정민정 | F      |    175 |     75 | 01059202038 | AB21     | 영창   |       3 |
|         4 | 김용연 | F      |    165 |     65 | 01090794634 | 49pD     | 영창   |    NULL |
|         5 | 김노연 | F      |    155 |     55 | 01024568059 | 6569     | 노량진 |       5 |
|         6 | 김민우 | M      |    175 |     92 | 01002942038 | an96     | 영창   |       6 |
|         7 | 최인호 | M      |    180 |     68 | 01089072769 | 0qi3     | 수지   |    NULL |
|         8 | 이진호 | M      |    184 |     74 | 01082937912 | 1298     | 노량진 |       5 |
|         9 | 권아영 | F      |    163 |     51 | 01049817291 | 59S2     | 노량진 |       5 |
|        10 | 성지윤 | F      |    153 |     48 | 01059197662 | va12     | 노량진 |       4 |
|        11 | 강지훈 | M      |    171 |     93 | 01098781613 | 09iS     | 수지   |       1 |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
11 rows in set (0.00 sec)
```

- DB2022_enroll : 7개

```
mysql> select * from DB2022_enroll;
+-----------+-------------+------------+------------+
| member_id | enroll_date | start_date | membership |
+-----------+-------------+------------+------------+
|         1 | 2022-05-22  | 2022-05-22 |          3 |
|         1 | 2022-05-22  | 2022-09-01 |          1 |
|         2 | 2022-05-10  | 2022-05-23 |          1 |
|         3 | 2021-04-19  | 2022-05-30 |          2 |
|         3 | 2021-05-20  | 2022-08-01 |          2 |
|         4 | 2018-05-01  | 2018-05-03 |          4 |
|         5 | 2022-05-23  | 2022-06-06 |          1 |
+-----------+-------------+------------+------------+
7 rows in set (0.00 sec)
```

3. 기본 키(primary key), 외래 키(foreign key), not null 제약조건(not null constraints)를 포함해야 한다 (빨간 밑줄)

```sql
create table DB2022_membership(
  membership_id int,
  name varchar(30) not null,
  period int not null,
  price int not null,

  primary key (membership_id)
);

create table DB2022_branches (
  name varchar(20),
  phone varchar(15) not null,
  city varchar(10) not null,
  state varchar(10) not null,
  street_name varchar(30) not null,
  street_number varchar(10) not null,
  boss varchar(20) not null,

  primary key (name)
);

create table DB2022_trainers (
  trainer_id int,
  name varchar(20) not null,
  career_start_year int not null,
  career_start_month int not null,
  branch varchar(30) not null,
  password varchar(4) not null,

  primary key (trainer_id),
  foreign key (branch) references DB2022_branches(name)
);
```

```sql
create table DB2022_members(
  member_id int,
  name varchar(20) not null,
  gender char(1) not null,
  height int not null,
  weight int not null,
  phone char(11) not null,
  password varchar(4) not null,
  branch varchar(20) not null,
  trainer int,

  primary key (member_id),
  foreign key (branch) references DB2022_branches(name),
  foreign key (trainer) references DB2022_trainers(trainer_id)
);

create table DB2022_enroll (
  member_id int,
  enroll_date date not null,
  start_date date,
  membership int not null,

  primary key (member_id, start_date), -- composite key
  foreign key (membership) references DB2022_membership(membership_id),
  foreign key (member_id) references DB2022_members(member_id)
);
```

4. View

```sql
/* 회원의 회원권 기간 */
create view DB2022_period as
select enroll.member_id, members.name, start_date, date_add(enroll.start_date, interval membership.period day) as en
d_date, membership.name as membership
from DB2022_enroll as enroll, DB2022_membership as membership, DB2022_members as members
where enroll.member_id=members.member_id and enroll.membership=membership.membership_id;

/* 트레이너 경력 */
create view DB2022_career as
select trainer_id,
floor(((year(now())-trainers.career_start_year)*12-trainers.career_start_month+month(now()))/12) as career_year,
mod((year(now())-trainers.career_start_year)*12-trainers.career_start_month+month(now()),12) as career_month
from DB2022_trainers as trainers;
```

DB2022_period

- 회원의 회원권 기간 정보를 담은 view이다.
- 회원 아이디, 회원 이름, 회원권 등록 날짜(회원권 구매 날짜), 시작 날짜(회원권을 사용하기 시작한 날짜), 만료 날짜, 회원권 이름 attribute를 포함한다.
- 마이페이지에서 사용자에게 자신이 보유한 회원권 기간을 알려주기 위해 사용된다.

DB2022_career

- 트레이너 경력 정보를 담은 view이다.
- 트레이너 아이디, (00년 00개월'로 표시할) 경력 연도, 경력 개월 attribute를 포함한다.
- 사용자가 PT를 신청하거나 트레이너 변경 시, 트레이너 경력을 제공하여 트레이너 선택에 도움을 준다.

5. 모든 테이블과 뷰의 이름들은 “DB2022_”라는 접두어를 가지고 있어야 한다.

3번과 4번을 통해 확인할 수 있다.

6. 적어도 4개의 인덱스를 정의해야 한다.

```
/* 트레이너들 지점별로 인덱싱 */
create index idx_trainer_branch
on DB2022_trainers(branch);
show index from DB2022_trainers;

/* 회원들 트레이너별로 인덱싱 */
create index idx_memb_trainer
on DB2022_members(trainer);
show index from DB2022_members;

/* 활동 회원들 회원권으로 인덱싱 */
create index idx_enroll_membership
on DB2022_enroll(membership);
show index from DB2022_enroll;

/* 회원들 번호별로 인덱싱 */
create unique index idx_memb_phone
on DB2022_members(phone);
show index from DB2022_members;
```

- **idx_trainer_branch :** 지점 별로 트레이너를 검색한다.
- **idx_memb_trainer :** 트레이너 별로 회원을 검색한다.
- **idx_enroll_membership :** 회원권 별로 활동 회원을 검색한다.
- **idx_memb_phone :** 휴대폰 번호 별로 회원을 검색한다.

7. 인덱스를 사용하는 쿼리들을 포함해야 한다.

(1) idx_trainer_branch

트레이너 관련 서비스를 제공하기 전에 시행되는 트레이너의 본인 확인 단계에서 사용된다.
DB2022_trainers table을 branch로 스캔하여 입력한 이름, 지점, 비밀번호와 DB2022_trainers의 name, branch, password가 같은 레코드를 검색하므로, 검색 속도를 높일 수 있다.

```
// check the trainer information
    identify_btn.addActionListener(new ActionListener() {

      @Override
      public void actionPerformed(ActionEvent e) {
        String name = name_field.getText().trim() == null ? "" : name_field.getText().trim();
        String branch = branch_field.getText().trim() == null ? "" : branch_field.getText().trim();
        String passwd = passwd_field.getText().trim() == null ? "" : passwd_field.getText().trim();

        try (Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBID, DB2022Team07_main.USERID,
            DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();) {
          PreparedStatement pStmt = conn.prepareStatement(
              "select trainer_id, branch from DB2022_trainers use index(idx_trainer_branch) where name=? and branch
=? and password=?;");
          pStmt.setString(1, name);
          pStmt.setString(2, branch);
          pStmt.setString(3, passwd);
          ResultSet rset = pStmt.executeQuery();
          // CASE 1: User enters the wrong trainer information
          if (!rset.next()) {
            t1.setText("<html><body style='text-align:center;'>"
                + "-----------------------------------------------------------------------------------<br/>"
                + "Wrong trainer Information!<br/>" + "Please enter it Again.<br/>"
                + "-----------------------------------------------------------------------------------"
                + "</body></html>");
          } else {
```

```
            // CASE 2: There is a trainer that matches the information.
            trainer_id = rset.getString(1);
            trainer_branch = rset.getString(2);

            quit_menu.setVisible(true);
            move_menu.setVisible(true);
            pwd_menu.setVisible(true);
            // print a list of service menus.
            t1.setText("<html><body style='text-align:center;'>"
                + "--------------------------------------------------------------------------------<br/>"
                + "Please select the service.<br/>"
                + "--------------------------------------------------------------------------------<br/>"
                + "</body></html>");
          }
          // sql error
        } catch (SQLException sqle) {
          System.out.println("SQLException: " + sqle);
        }

      }

    });
```

(2) idx_memb_trainer

- 지점 별 누적 회원 수 TOP1 트레이너를 보여주는 서비스에서 사용된다. (트레이너가 지점을 이동할 경우, 해당 트레이너의 누적 회원 수는 0명이 된다.)
  DB2022_members table을 trainer로 스캔하여 DB2022_members.trainer = DB2022.trainers.trainer_id인 레코드를 검색하므로, 검색 속도를 높일 수 있다.

```
// 버튼에 지점별 누적 회원 수 1위 트레이너 보여주는 이벤트 추가
      top1_btn.addActionListener(new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
             try(Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DB
ID,
                   DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();){
                ResultSet rset = stmt.executeQuery("with A as "
                                + "(select trainer, DB2022_trainers.branch, DB2022_trainers.name, count(trainer) as
 totalMemb "
                                + "from DB2022_trainers, DB2022_members use index(idx_memb_trainer)"
                                  + "where DB2022_members.trainer = DB2022_trainers.trainer_id group by trainer), \r
\n"
                                  + "B as "
                                  + "(select branch, max(totalMemb) as topMemb "
                                  + "from A group by branch) "
                                  + "select A.branch, name, topMemb "
                                  + "from A,B "
                                  + "where A.branch = B.branch and totalMemb = topMemb order by branch;");
                top1_label.setText("<html><body style='text-align:center;'>");
                      while (rset.next()) {
                         String str = (rset.getString("branch") + ": " + rset.getString("name")+ " trainer (" + rse
t.getInt(3)+ " members)" );
                         top1_label.setText(top1_label.getText() + str+"<br/>");
                         System.out.println(str);
                      }
                      top1_label.setText(top1_label.getText()+"----------------------------------------------------
----------------------------<br/></body></html>");
             } catch(SQLException sqle) {
                System.out.println("SQL Exception: "+sqle);
             }
          }
       });
```

- 지점을 이동하는 트레이너가 담당한 회원의 DB2022_members.trainer field를 null로 변경하는 과정에서 사용된다.
  DB2022_members table을 trainer로 스캔하여 DB2022_members.trainer와 지점을 이동하는 트레이너의 id값이 같은 레코드를 검색하므로, 검색 속도를 높일 수 있다.

```
void Trainer_Move(String trainer, String branch) {
    try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBID,
        DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();) {
      if (branch_list.contains(branch)) {
        // if user entered the existing branch name
        try {
          // transaction
          conn.setAutoCommit(false);
          // members assigned by the trainer: change the trainer field to null.
          PreparedStatement pStmt = conn
              .prepareStatement("update DB2022_members use index(idx_memb_trainer) set trainer = null where trainer
=?;");
          pStmt.setString(1, trainer);
          pStmt.executeUpdate();
          // change the trainer's branch
          pStmt = conn.prepareStatement("update DB2022_trainers set branch =? where trainer_id=?;");
          pStmt.setString(1, branch);
          pStmt.setString(2, trainer);
          pStmt.executeUpdate();
          conn.commit();
          t4.setText("<html><body style='text-align:center;'>"
              + "-----------------------------------------------------------------------------------<br/>"
              + "Branch is changed.<br/>"
              + "-----------------------------------------------------------------------------------<br/>"
              + "</body></html>");
          trainer_branch = branch;
          close.setVisible(true);
        } catch (SQLException se) {
          // handle error for JDBC
          se.printStackTrace();
          System.out.println("Transaction Error");
          try {
            // if there is an error then rollback the commit.
            if (conn != null)
              conn.rollback();
          } catch (SQLException se2) {
            se2.printStackTrace();
          } // end try

        }
        conn.setAutoCommit(true);
      } else {
        // if user entered the wrong branch name
        t4.setText("<html><body style='text-align:center;'>"
            + "-----------------------------------------------------------------------------------<br/>"
            + "Wrong branch name!<br/>" + "Please re-enter the branch name.<br />"
            + "-----------------------------------------------------------------------------------<br/>"
            + "</body></html>");
      }
    } catch (SQLException sqle) {
      System.out.println("SQLException: " + sqle);
    }

  }
```

- 퇴사하는 트레이너가 담당한 회원의 DB2022_members.trainer field를 null로 변경하는 과정에서 사용된다. DB2022_members table을 trainer로 스캔하여 DB2022_members.trainer와 퇴사하는 트레이너의 id값이 같은 레코드를 검색하므로, 검색 속도를 높일 수 있다.

```
void Trainer_Quit(String trainer) {
    try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBID,
        DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();) {
      try {
        // transaction
        conn.setAutoCommit(false);
        // members assigned by the trainer: change the trainer field to null.
        PreparedStatement pStmt = conn
            .prepareStatement("update DB2022_members use index(idx_memb_trainer) set trainer = null where trainer
=?;");
        pStmt.setString(1, trainer);
        pStmt.executeUpdate();
        // resign the trainer
```

```
        pStmt = conn.prepareStatement("delete from DB2022_trainers where trainer_id=?;");
        pStmt.setString(1, trainer);
        pStmt.executeUpdate();
        conn.commit();
        t2.setText("<html><body style='text-align:center;'>"
            + "-----------------------------------------------------------------------------<br/>"
            + "Resignation has been processed.<br/>"
            + "-----------------------------------------------------------------------------<br/>"
            + "</body></html>");
        close.setVisible(true);
      } catch (SQLException se) {
        // handle error for JDBC
        se.printStackTrace();
        System.out.println("Transaction Error");
        try {
          // if there is an error, then rollback the commit.
          if (conn != null)
            conn.rollback();
        } catch (SQLException se2) {
          se2.printStackTrace();
        } // end try
      }
      conn.setAutoCommit(true);
      move_menu.setEnabled(false);
      pwd_menu.setEnabled(false);

    } catch (SQLException sqle) {
      System.out.println("SQLException: " + sqle);
    }

  }
```

(3) idx_enroll_membership

지점 별 월 매출을 계산하는 서비스에서 사용된다.

DB2022_enroll table을 membership으로 스캔하여 DB2022_enroll.membership = DB2022_membership.membership_id인 레코드를 join하므로, join 과정의 레코드 스캔 시간을 줄일 수 있다.

```
// 올바른 값을 입력했으면 월 매출 계산하기
      if (year != -1 && month != -1) {
         try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBID,
               DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();) {
            PreparedStatement pStmt = conn.prepareStatement(
                  "select branch, sum(price) as 'revenue' " + "from DB2022_enroll use index(idx_enroll_membership)jo
in DB2022_membership "
                        + "on DB2022_enroll.membership=DB2022_membership.membership_id "
                        + "join DB2022_members using(member_id) "
                        + "where year(enroll_date)=? and month(enroll_date)=? " + "group by branch "
                        + "order by branch;");
            pStmt.setString(1, tempYear);
            pStmt.setString(2, tempMonth);
            ResultSet rset = pStmt.executeQuery();
            // 월별 매출 보여주기
            if (rset.next()) {
               String info;
               profit.setText("<html><body style='text-align:center;'>"
                     + "-----------------------------------------------------------------------------<br/>");
               do {
                  info = rset.getString("branch") + ": " + rset.getInt("revenue") + " won<br/>";
                  profit.setText(profit.getText() + info);
               } while (rset.next());
               profit.setText(profit.getText() + "</body></html>");
            } else {
              // 해당 날짜에 매출 정보가 없으면 정보가 없다는 메시지 띄우기
               profit.setText("<html><body style='text-align:center;'>"
                     + "-----------------------------------------------------------------------------<br/>"
                     + "There are no members registered in the year and month..<br/>"
                     + "-----------------------------------------------------------------------------<br/>"
                     + "</body></html>");
            }

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

```
        }
    }
```

(4) idx_memb_phone (unique index)

회원권 구매 서비스, 트레이너 등록/변경 서비스, BMI 조회 서비스, 회원권 만료 날짜 조회 서비스, 회원 정보 수정 서비스를 제공하기 전에 시행되는 회원의 본인 확인 단계에서 사용된다.
DB2022_members table을 phone으로 스캔하여 DB2022_members.phone과 입력한 전화번호가 일치하는 레코드를 검색하므로, 검색 속도를 높일 수 있다.

```
// 회원권 구매 서비스 제공 전 시행되는 본인 확인 코드
PreparedStatement pStmt = conn.prepareStatement("select * from DB2022_members use index(idx_memb_phone) where name=?
and phone=?;");
      pStmt.setString(1, member_name);
      pStmt.setString(2, phone);
      ResultSet info_set = pStmt.executeQuery();
// BMI 조회 서비스, 회원권 만료 날짜 조회 서비스, 회원 정보 수정 서비스 제공 전 시행되는 본인 환인 코드
PreparedStatement pStmt = conn
              .prepareStatement("select membership, height, weight, phone, end_date "
                  + "from DB2022_members use index(idx_memb_phone) join DB2022_period using(member_id) "
                  + "where DB2022_members.name=? and phone=? and password=?;");
          pStmt.setString(1, name);
          pStmt.setString(2, phone);
          pStmt.setString(3, passwd);
          ResultSet rset = pStmt.executeQuery();
// 트레이너 등록/변경 서비스 제공 전 시행되는 본인 확인 코드
PreparedStatement pStmt = conn.prepareStatement(
          "select * from DB2022_members use index(idx_memb_phone) join DB2022_enroll using(member_id) where name=? a
nd phone=?;");
      pStmt.setString(1, member_name);
      pStmt.setString(2, phone);
      ResultSet info_set = pStmt.executeQuery();
```

회원 가입 서비스에서 DB2022_members table의 primary key는 자동 생성되므로 중복된 회원이 등록될 수 있다.
따라서 DB2022_members.phone을 unique index로 설정하여, 전화번호가 같은 회원은 중복 등록될 수 없도록 설정하였다.

8. 사용자가 뷰를 사용하는 쿼리를 포함해야 한다.

DB2022_career view

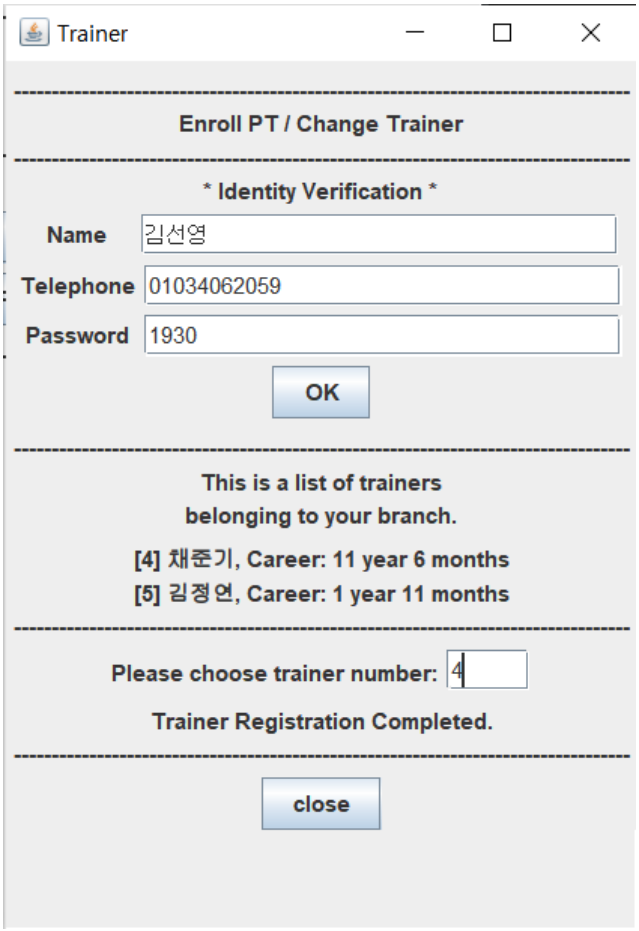
: 사용자가 등록 혹은 변경할 트레이너를 선택하는 과정에서 트레이너의 경력을 참고하므로, 트레이너의 경력을 출력하는 query에서 해당 뷰를 사용한다

TrainerFrame class의 Trainer_Change 메소드에서 DB2022_career view가 사용된 코드이다.

```
t2.setText("<html><body style='text-align:center;'>");
          while (trainer_set.next()) {
            id.add(trainer_set.getInt("trainer_id"));
            t2.setText(t2.getText() + "[" + trainer_set.getString("trainer_id")+"]" + " "
                + trainer_set.getString("name") + ", Career: " + trainer_set.getString("career_year") + " year "
                + trainer_set.getString("career_month") + " months<br />");

          }
          t2.setText(t2.getText() + "----------------------------------------------------------------------------------
---</body></html>");
          t3.setText("<html><body style='text-align:center;'>Please choose trainer number: </body></html>");
          trainer_field.setVisible(true);
```

트레이너 등록/변경 서비스가 실행된 결과 화면이다. “This is a list of trainers belonging to your branch.” 아래에 트레이너의 경력이 출력된 것을 볼 수 있다.



DB2022_period

: 회원이 본인이 사용중인 회원권의 만료 날짜를 조회하거나, 관리자가 회원권이 만료된 회원의 등록 정보를 삭제하는 query에서 해당 뷰를 사용한다.

1. **MyPageFrame class에서 DB2022_period view가 사용된 코드이다.**

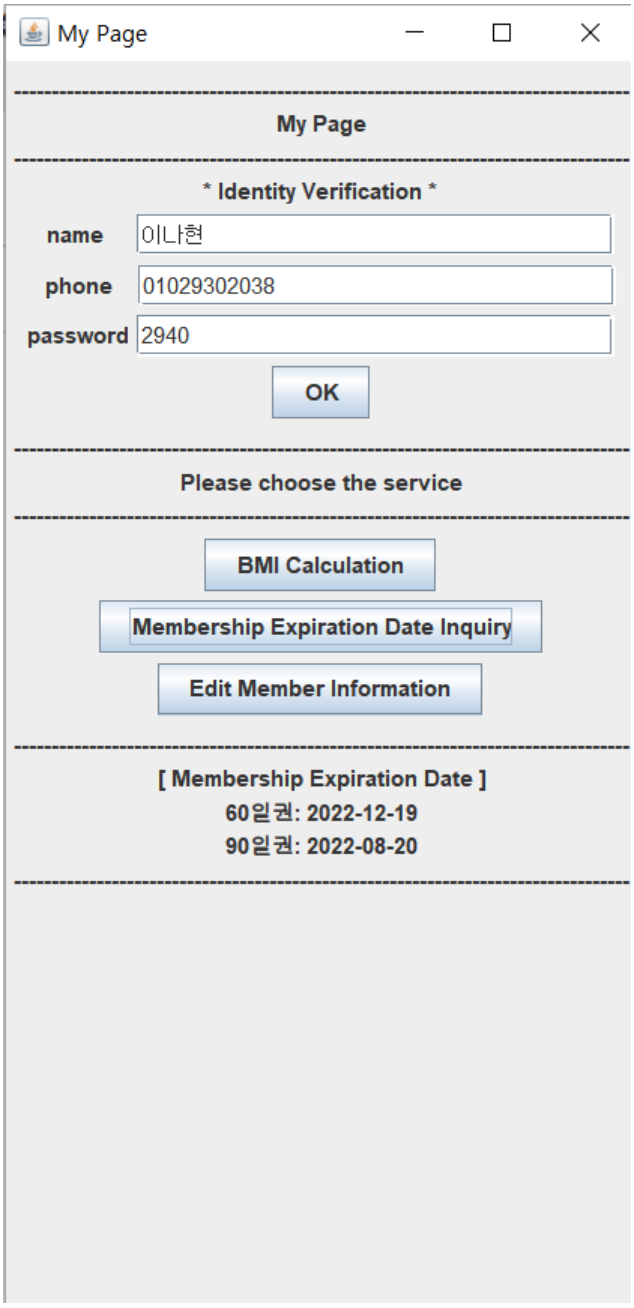
end_date_list vector에 조회한 end_date와 membership을 저장한 다음, end_date_list에 저장된 문자열을 차례로 출력한다.

```
PreparedStatement pStmt = conn.prepareStatement(
              "select membership, height, weight, phone, end_date "
              + "from DB2022_members join DB2022_period using(member_id) "
              + "where DB2022_members.name=? and phone=? and password=?;");
          pStmt.setString(1, name);
          pStmt.setString(2, phone);
          pStmt.setString(3, passwd);
          ResultSet rset = pStmt.executeQuery();

          if (!rset.next()) {
            t1.setText("<html><body style='text-align:center;'>"
                + "-----------------------------------------------------------------------------<br/>"
                + "Invalid member information.<br/>" + "Please re-enter your member information.<br/>"
                + "-----------------------------------------------------------------------------"
                + "</body></html>");
          } else {
            height = rset.getString("height");
            weight = rset.getString("weight");
            phone = rset.getString("phone");
            do {
              end_date_list.add(rset.getString("membership")+": "+rset.getString("end_date"));
            } while (rset.next());
```

```
t2.setText("<html><body style='text-align:center;'>"
            + "-----------------------------------------------------------------------------------------<br/>"
            + "[ Membership Expiration Date ]<br/>");
        while(!end_date_list.isEmpty()) {
          t2.setText(t2.getText()+end_date_list.lastElement()+"<br/>");
          end_date_list.remove(end_date_list.size()-1);
        }
        t2.setText(t2.getText()
            + "-----------------------------------------------------------------------------------------<br/>"
            + "</body></html>");
```

아래는 회원이 본인이 사용 중인 회원권의 만료 날짜를 조회한 결과 화면이다. 회원권의 종류와 만료 날짜가 출력되는 것

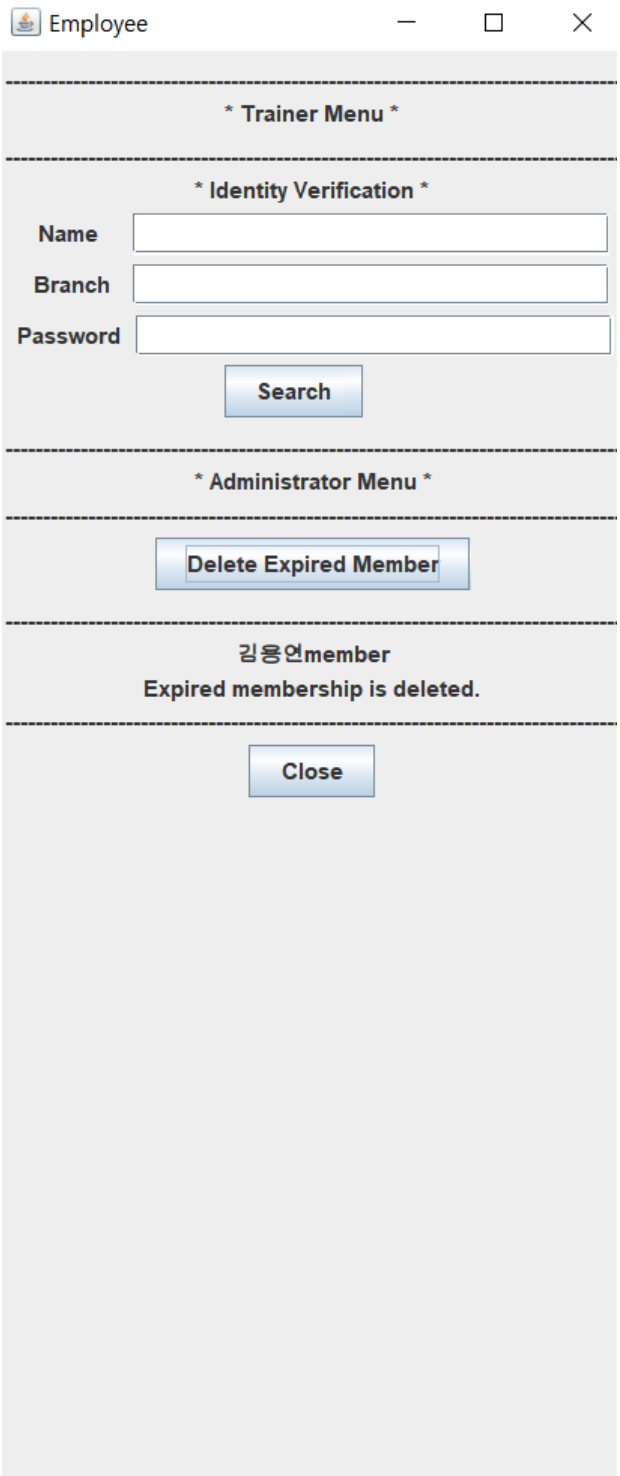


2. **<u>EmployeeFrame class에서 DB2022_period view가 사용된 코드이다.</u>**

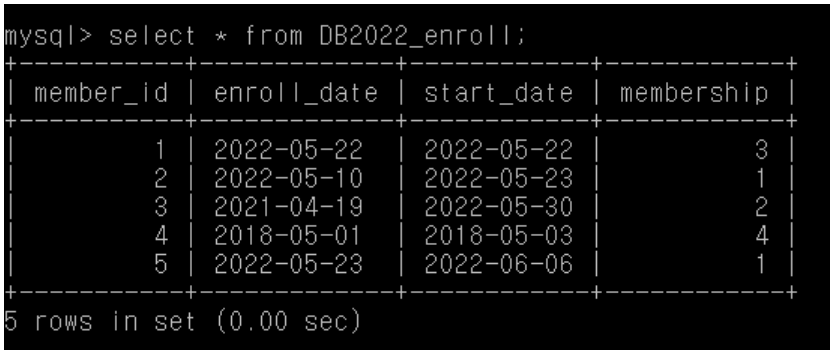
   현재 날짜와 DB2022_period.end_date를 연산하여 만료된 회원권 정보를 구별한다.

```
ResultSet rset = stmt.executeQuery(
        "select name,member_id from DB2022_period where timestampdiff(day,end_date,curdate())>0;");
    // if there is no expired member
    if (!rset.next()) {
      t5.setText("<html><body style='text-align:center;'>"
          + "-----------------------------------------------------------------------------------<br/>"
          + "There is no expired member.<br/>"
          + "-----------------------------------------------------------------------------------"
          + "</body></html>");
    } else {
      // if there are expired members
      t5.setText("<html><body style='text-align:center;'>"
          + "-----------------------------------------------------------------------------------<br/>");
      PreparedStatement pStmt;
      do {
        // print a list of expired members
        t5.setText(t5.getText() + rset.getString("name") + "member<br/>");
        // delete expired members from DB2022_enroll table (= convert expired member to rest member)
        pStmt = conn.prepareStatement("delete from DB2022_enroll where member_id=?");
        pStmt.setString(1, rset.getString("member_id"));
        pStmt.executeUpdate();
      } while (rset.next());
      t5.setText(t5.getText() + "Expired members has been converted to rest members.<br/>"
          + "-----------------------------------------------------------------------------------"
          + "</body></html>");
    }
```

아래는 관리자가 만료된 회원권 정보를 삭제한 결과 화면이다. 만료된 회원 정보가 뜨며 회원권 정보가 삭제되는 것을 알 수 있다.

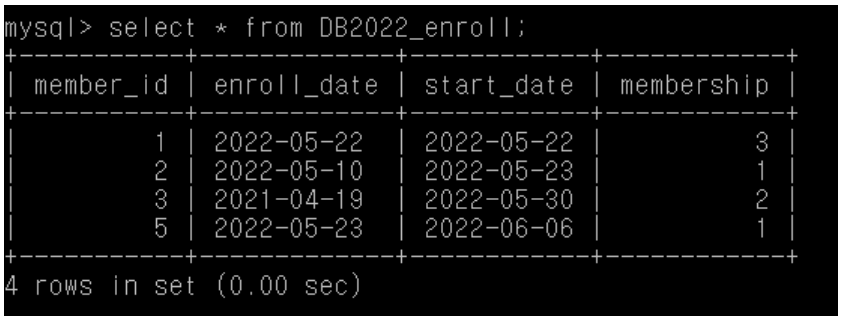


아래는 Mysql으로 데이터베이스 상태를 비교한 결과 화면이다. 만료된 회원권 정보가 DB2022_enroll에서 삭제된 것을 알 수 있다.



```
mysql> select * from DB2022_enroll;
+-----------+-------------+------------+------------+
| member_id | enroll_date | start_date | membership |
+-----------+-------------+------------+------------+
|         1 | 2022-05-22  | 2022-05-22 |          3 |
|         2 | 2022-05-10  | 2022-05-23 |          1 |
|         3 | 2021-04-19  | 2022-05-30 |          2 |
|         4 | 2018-05-01  | 2018-05-03 |          4 |
|         5 | 2022-05-23  | 2022-06-06 |          1 |
+-----------+-------------+------------+------------+
5 rows in set (0.00 sec)
```

서비스 실행 전 데이터베이스 상태



```
mysql> select * from DB2022_enroll;
+-----------+-------------+------------+------------+
| member_id | enroll_date | start_date | membership |
+-----------+-------------+------------+------------+
|         1 | 2022-05-22  | 2022-05-22 |          3 |
|         2 | 2022-05-10  | 2022-05-23 |          1 |
|         3 | 2021-04-19  | 2022-05-30 |          2 |
|         5 | 2022-05-23  | 2022-06-06 |          1 |
+-----------+-------------+------------+------------+
4 rows in set (0.00 sec)
```

서비스 실행 후 데이터베이스 상태

만료된 회원권 정보가 삭제된 것을 알 수 있다.

9. 트랜잭션(transaction)을 포함해야 한다.

트레이너 지점 이동 신청

트레이너가 담당한 회원의 DB2022_members.trainer를 null로 변경한 후, 트레이너의DB2022_trainers.branch를 변경한다. 2개의 query를 Transaction으로 설정하여 트레이너가 담당한 회원의 trainer field만 null로 변경되거나 트레이너의 지점만 변경되는 상황을 예방한다.

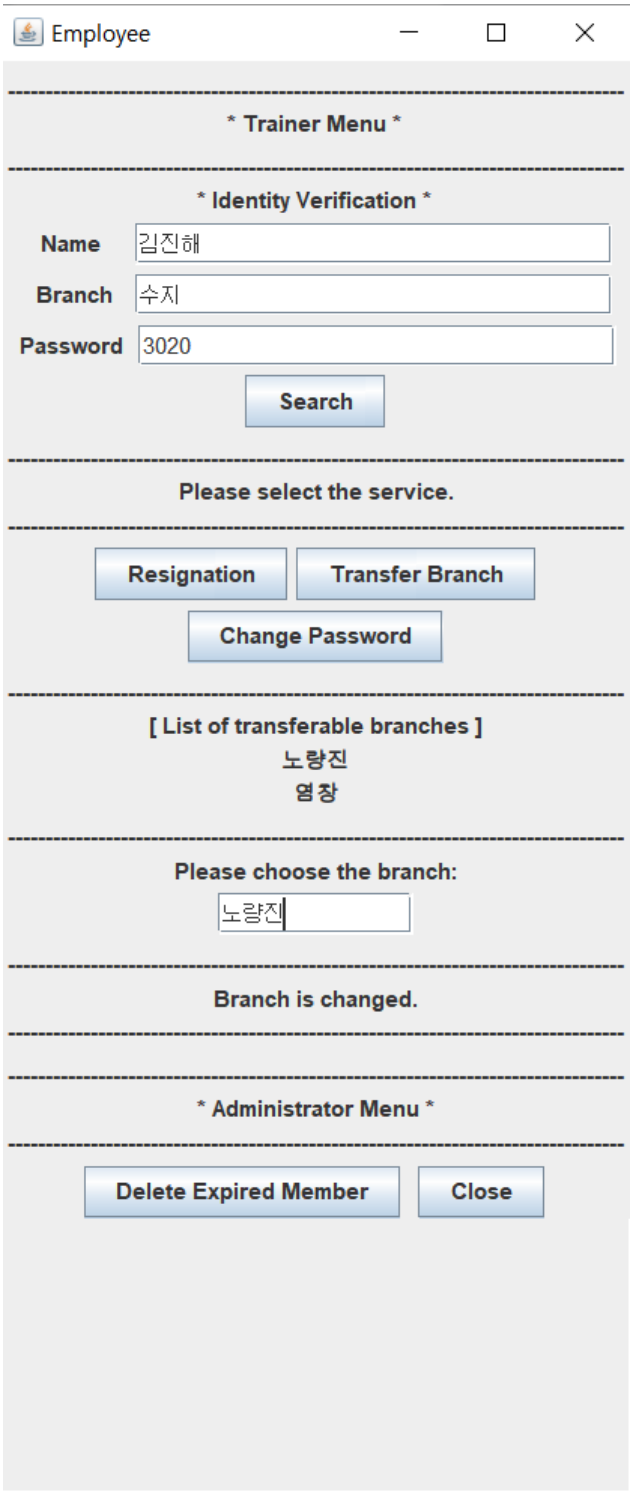
아래는 EmployeeFrame class의 Trainer_Move 메소드에서 Transaction이 사용된 코드이다.

```
try {
          // transaction
          conn.setAutoCommit(false);
          // members assigned by the trainer: change the trainer field to null.
          PreparedStatement pStmt = conn
              .prepareStatement("update DB2022_members set trainer = null where trainer=?;");
          pStmt.setString(1, trainer);
          pStmt.executeUpdate();
          // change the trainer's branch
          pStmt = conn.prepareStatement("update DB2022_trainers set branch =? where trainer_id=?;");
          pStmt.setString(1, branch);
          pStmt.setString(2, trainer);
          pStmt.executeUpdate();
          conn.commit();
          t4.setText("<html><body style='text-align:center;'>"
              + "-----------------------------------------------------------------------------<br/>"
              + "Branch is changed.<br/>"
              + "-----------------------------------------------------------------------------<br/>"
              + "</body></html>");
```

```
            trainer_branch = branch;
            close.setVisible(true);
          } catch (SQLException se) {
            // handle error for JDBC
            se.printStackTrace();
            System.out.println("Transaction Error");
            try {
              // if there is an error then rollback the commit.
              if (conn != null)
                conn.rollback();
            } catch (SQLException se2) {
              se2.printStackTrace();
            } // end try

          }
          conn.setAutoCommit(true);
```

아래는 트레이너 지점 이동 신청 서비스가 완료된 결과 화면이다.



아래는 Mysql으로 데이터베이스 상태를 비교한 결과 화면이다.

해당 트레이너가 담당한 회원의 DB2022_members.trainer가 null로 변경되고, 해당 트레이너의 DB2022_trainers.branch가 변경되었음을 알 수 있다.

```
mysql> select * from DB2022_MEMBERS;
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
| member_id | name   | gender | height | weight | phone       | password | branch | trainer |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
|         1 | 이나현 | F      |    160 |     60 | 01029302038 | 2940     | 수지   |       1 |
|         2 | 김선영 | F      |    170 |     70 | 01034062059 | 1930     | 노량진 |       4 |
|         3 | 정민정 | F      |    175 |     75 | 01059202038 | 4021     | 염창   |       3 |
|         4 | 김용연 | F      |    165 |     65 | 01090794634 | 4920     | 염창   |    NULL |
|         5 | 김도연 | F      |    155 |     55 | 01024568059 | 6569     | 노량진 |       5 |
|         6 | 김민우 | M      |    175 |     92 | 01002942038 | 4496     | 염창   |       6 |
|         7 | 최인호 | M      |    180 |     68 | 01089072769 | 0123     | 수지   |    NULL |
|         8 | 이진호 | M      |    184 |     74 | 01082937912 | 1298     | 노량진 |       5 |
|         9 | 권아영 | F      |    163 |     51 | 01049817291 | 5902     | 노량진 |       5 |
|        10 | 성지윤 | F      |    153 |     48 | 01059197662 | 4912     | 노량진 |       4 |
|        11 | 강지훈 | M      |    171 |     93 | 01098781613 | 0913     | 수지   |       1 |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
11 rows in set (0.00 sec)

mysql> select * from DB2022_TRAINERS;
+------------+--------+-------------------+--------------------+--------+----------+
| trainer_id | name   | career_start_year | career_start_month | branch | password |
+------------+--------+-------------------+--------------------+--------+----------+
|          1 | 김진해 |              2017 |                  5 | 수지   | 3020     |
|          2 | 정완섭 |              2022 |                  6 | 수지   | 5860     |
|          3 | 도실인 |              2013 |                 11 | 염창   | 3068     |
|          4 | 채준기 |              2010 |                 12 | 노량진 | 6039     |
|          5 | 김정연 |              2020 |                  7 | 노량진 | 1059     |
|          6 | 이은혜 |              2015 |                  9 | 염창   | 7049     |
+------------+--------+-------------------+--------------------+--------+----------+
6 rows in set (0.00 sec)
```

서비스 실행 전 데이터베이스 상태

1번 트레이너의 지점이 “노량진”으로 변경되고, 1번 트레이너가 담당한 회원이 존재하지 않는다.

```
mysql> select * from DB2022_MEMBERS;
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
| member_id | name   | gender | height | weight | phone       | password | branch | trainer |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
|         1 | 이나현 | F      |    160 |     60 | 01029302038 | 2940     | 수지   |    NULL |
|         2 | 김선영 | F      |    170 |     70 | 01034062059 | 1930     | 노량진 |       4 |
|         3 | 정민정 | F      |    175 |     75 | 01059202038 | 4021     | 염창   |       3 |
|         4 | 김용연 | F      |    165 |     65 | 01090794634 | 4920     | 염창   |    NULL |
|         5 | 김도연 | F      |    155 |     55 | 01024568059 | 6569     | 노량진 |       5 |
|         6 | 김민우 | M      |    175 |     92 | 01002942038 | 4496     | 염창   |       6 |
|         7 | 최인호 | M      |    180 |     68 | 01089072769 | 0123     | 수지   |    NULL |
|         8 | 이진호 | M      |    184 |     74 | 01082937912 | 1298     | 노량진 |       5 |
|         9 | 권아영 | F      |    163 |     51 | 01049817291 | 5902     | 노량진 |       5 |
|        10 | 성지윤 | F      |    153 |     48 | 01059197662 | 4912     | 노량진 |       4 |
|        11 | 강지훈 | M      |    171 |     93 | 01098781613 | 0913     | 수지   |    NULL |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
11 rows in set (0.00 sec)

mysql> select * from DB2022_TRAINERS;
+------------+--------+-------------------+--------------------+--------+----------+
| trainer_id | name   | career_start_year | career_start_month | branch | password |
+------------+--------+-------------------+--------------------+--------+----------+
|          1 | 김진해 |              2017 |                  5 | 노량진 | 3020     |
|          2 | 정완섭 |              2022 |                  6 | 수지   | 5860     |
|          3 | 도실인 |              2013 |                 11 | 염창   | 3068     |
|          4 | 채준기 |              2010 |                 12 | 노량진 | 6039     |
|          5 | 김정연 |              2020 |                  7 | 노량진 | 1059     |
|          6 | 이은혜 |              2015 |                  9 | 염창   | 7049     |
+------------+--------+-------------------+--------------------+--------+----------+
```

서비스 실행 후 데이터베이스 상태

트레이너 퇴사 신청

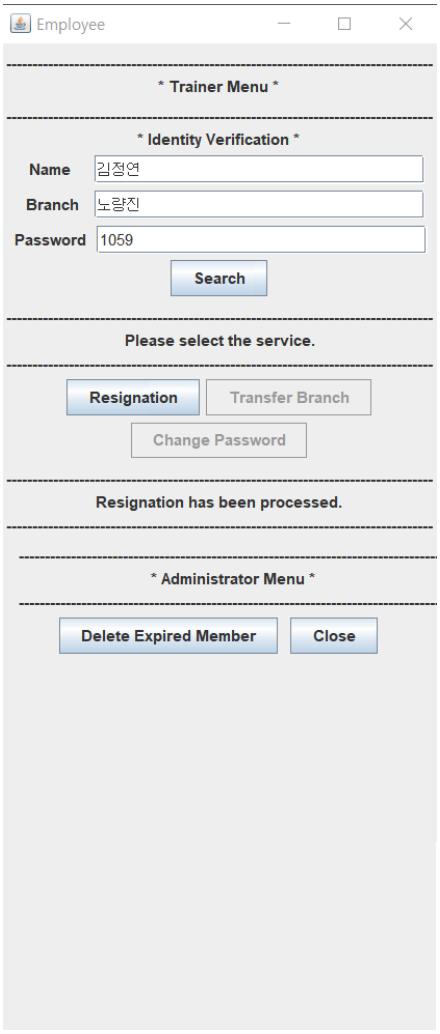
퇴사한 트레이너가 담당한 회원의 DB2022_members.trainer를 null로 변경한 후, DB2022_trainers에서 트레이너 정보를 삭제한다.

2개의 query를 Transaction으로 설정하여 퇴사한 트레이너가 담당한 회원의 trainer field만 null로 변경되거나 트레이너 정보만 삭제되는 상황을 예방한다.

아래는 EmployeeFrame class의 Trainer_Quit 메소드에서 Transaction이 사용된 코드이다.

```
276             try {
277                 // transaction
278                 conn.setAutoCommit(false);
279                 // members assigned by the trainer: change the trainer field to null.
280                 PreparedStatement pStmt = conn
281                         .prepareStatement("update DB2022_members set trainer = null where trainer=?;");
282                 pStmt.setString(1, trainer);
283                 pStmt.executeUpdate();
284                 // resign the trainer
285                 pStmt = conn.prepareStatement("delete from DB2022_trainers where trainer_id=?;");
286                 pStmt.setString(1, trainer);
287                 pStmt.executeUpdate();
288                 conn.commit();
289                 t2.setText("<html><body style='text-align:center;'>"
290                         + "-----------------------------------------------------------------------------------------<br/>"
291                         + "Resignation has been processed.<br/>"
292                         + "-----------------------------------------------------------------------------------------<br/>"
293                         + "</body></html>");
294                 close.setVisible(true);
295             } catch (SQLException se) {
296                 // handle error for JDBC
297                 se.printStackTrace();
298                 System.out.println("Transaction Error");
299                 try {
300                     // if there is an error, then rollback the commit.
301                     if (conn != null)
302                         conn.rollback();
303                 } catch (SQLException se2) {
304                     se2.printStackTrace();
305                 } // end try
306             }
307             conn.setAutoCommit(true);
```

아래는 트레이너 퇴사 신청 서비스가 완료된 결과 화면이다.



아래는 Mysql으로 데이터베이스 상태를 비교한 결과 화면이다.

해당 트레이너가 담당한 회원의 DB2022_members.trainer가 null로 변경되고, DB2022_trainers에서 해당 트레이너 정보가 삭제되었음을 알 수 있다.

5번 트레이너가 존재하고, 5번 트레이너가 담당한 회원이 존재한다.

```
mysql> select * from DB2022_MEMBERS;
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
| member_id | name   | gender | height | weight | phone       | password | branch | trainer |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
|         1 | 이나현 | F      |    160 |     60 | 01029302038 | 2940     | 수지   |    NULL |
|         2 | 김선영 | F      |    170 |     70 | 01034062059 | 1930     | 노량진 |       4 |
|         3 | 정민정 | F      |    175 |     75 | 01059202038 | 4021     | 염창   |       3 |
|         4 | 김용연 | F      |    165 |     65 | 01090794634 | 4920     | 염창   |    NULL |
|         5 | 김도연 | F      |    155 |     55 | 01024568059 | 6569     | 노량진 |       5 |
|         6 | 김민우 | M      |    175 |     92 | 01002942038 | 4496     | 염창   |       6 |
|         7 | 최인호 | M      |    180 |     68 | 01089072769 | 0123     | 수지   |    NULL |
|         8 | 이진호 | M      |    184 |     74 | 01082937912 | 1298     | 노량진 |       5 |
|         9 | 권아영 | F      |    163 |     51 | 01049817291 | 5902     | 노량진 |       5 |
|        10 | 성지윤 | F      |    153 |     48 | 01059197662 | 4912     | 노량진 |       4 |
|        11 | 강지훈 | M      |    171 |     93 | 01098781613 | 0913     | 수지   |    NULL |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
11 rows in set (0.00 sec)

mysql> select * from DB2022_TRAINERS;
+------------+--------+-------------------+--------------------+--------+----------+
| trainer_id | name   | career_start_year | career_start_month | branch | password |
+------------+--------+-------------------+--------------------+--------+----------+
|          1 | 김진해 |              2017 |                  5 | 노량진 | 3020     |
|          2 | 정완섭 |              2022 |                  6 | 수지   | 5860     |
|          3 | 도실인 |              2013 |                 11 | 염창   | 3068     |
|          4 | 채준기 |              2010 |                 12 | 노량진 | 6039     |
|          5 | 김정연 |              2020 |                  7 | 노량진 | 1059     |
|          6 | 이은혜 |              2015 |                  9 | 염창   | 7049     |
+------------+--------+-------------------+--------------------+--------+----------+
```

서비스 실행 전 데이터베이스 상태

5번 트레이너가 삭제되고, 5번 트레이너가 담당한 회원이 존재하지 않는다.

```
mysql>  select * from DB2022_MEMBERS;
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
| member_id | name   | gender | height | weight | phone       | password | branch | trainer |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
|         1 | 이나현 | F      |    160 |     60 | 01029302038 | 2940     | 수지   |    NULL |
|         2 | 김선영 | F      |    170 |     70 | 01034062059 | 1930     | 노량진 |       4 |
|         3 | 정민정 | F      |    175 |     75 | 01059202038 | 4021     | 염창   |       3 |
|         4 | 김용연 | F      |    165 |     65 | 01090794634 | 4920     | 염창   |    NULL |
|         5 | 김도연 | F      |    155 |     55 | 01024568059 | 6569     | 노량진 |    NULL |
|         6 | 김민우 | M      |    175 |     92 | 01002942038 | 4496     | 염창   |       6 |
|         7 | 최인호 | M      |    180 |     68 | 01089072769 | 0123     | 수지   |    NULL |
|         8 | 이진호 | M      |    184 |     74 | 01082937912 | 1298     | 노량진 |    NULL |
|         9 | 권아영 | F      |    163 |     51 | 01049817291 | 5902     | 노량진 |    NULL |
|        10 | 성지윤 | F      |    153 |     48 | 01059197662 | 4912     | 노량진 |       4 |
|        11 | 강지훈 | M      |    171 |     93 | 01098781613 | 0913     | 수지   |    NULL |
+-----------+--------+--------+--------+--------+-------------+----------+--------+---------+
11 rows in set (0.01 sec)

mysql> select * from DB2022_TRAINERS;
+------------+--------+-------------------+--------------------+--------+----------+
| trainer_id | name   | career_start_year | career_start_month | branch | password |
+------------+--------+-------------------+--------------------+--------+----------+
|          1 | 김진해 |              2017 |                  5 | 노량진 | 3020     |
|          2 | 정완섭 |              2022 |                  6 | 수지   | 5860     |
|          3 | 도실인 |              2013 |                 11 | 염창   | 3068     |
|          4 | 채준기 |              2010 |                 12 | 노량진 | 6039     |
|          6 | 이은혜 |              2015 |                  9 | 염창   | 7049     |
+------------+--------+-------------------+--------------------+--------+----------+
5 rows in set (0.00 sec)
```

서비스 실행 후 데이터베이스 상태

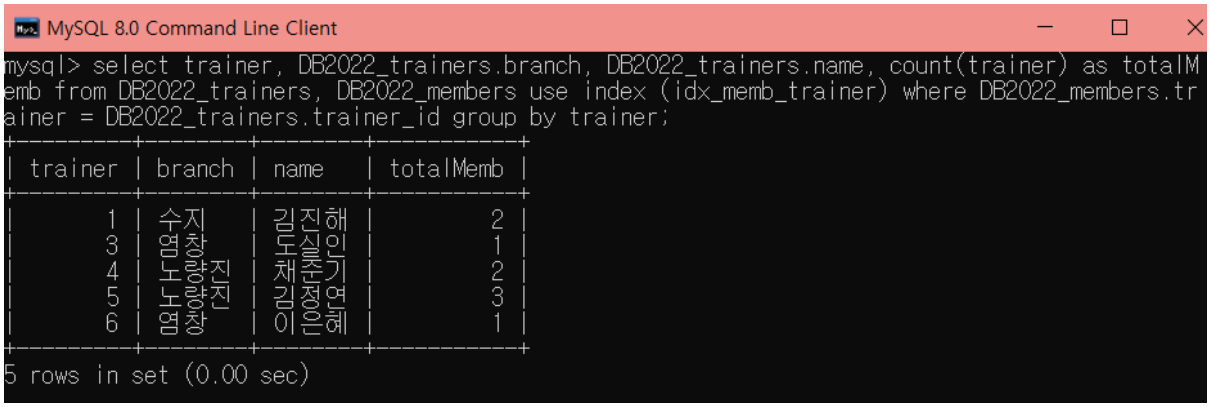
10. 중첩된 쿼리(nested query)들을 가지는 쿼리들을 포함해야 한다.

BranchFrame.java의 지점별 회원 수 1위 트레이너 보여주는 기능 관련 코드에 포함된 SQL문

```
with A as (select trainer, DB2022_trainers.branch, DB2022_trainers.name, count(trainer) as totalMemb
from DB2022_trainers, DB2022_members use index (idx_memb_trainer) where DB2022_members.trainer = DB2022_trainers.tra
iner_id group by trainer),
B as (select branch, max(totalMemb) as topMemb from A group by branch) select A.branch, name, topMemb
from A,B  where A.branch = B.branch and totalMemb = topMemb order by branch;
```

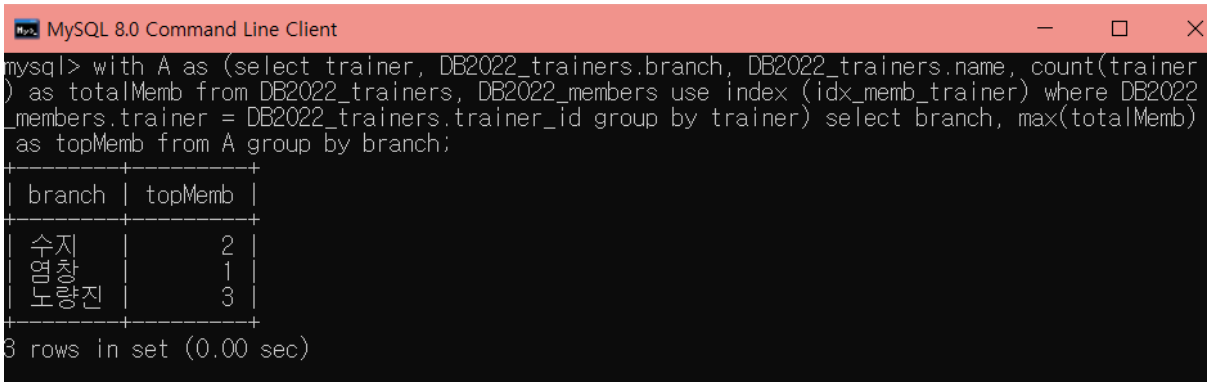
A 릴레이션

: 맡은 회원이 없는 트레이너를 제외한 모든 트레이너에 대해 해당 트레이너가 맡은 회원의 수 정보를 보여줌



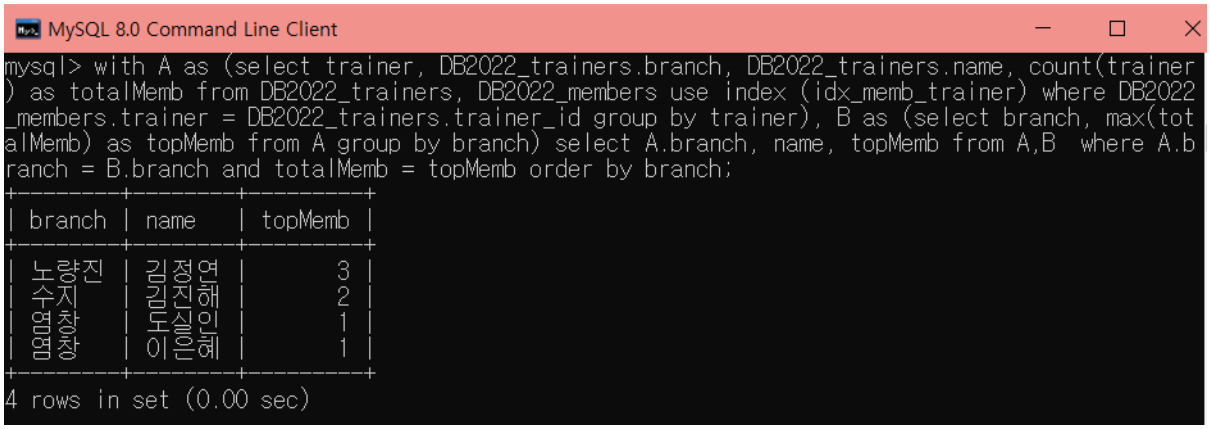
B 릴레이션

: 지점별로 트레이너가 맡은 최대 누적 회원 수 정보를 보여줌



최종 select 릴레이션

: A, B 릴레이션을 통해 최종적으로 얻고자 하는 정보를 보여줌



11. 조인 쿼리(join query)들을 가지는 쿼리들은 포함해야 한다.

```
-- BranchFrame
select branch, sum(price) as 'revenue' from DB2022_enroll join DB2022_membership
on DB2022_enroll.membership=DB2022_membership.membership_id join DB2022_members
using(member_id) where year(enroll_date)=? and month(enroll_date)=?
group by branch order by branch;
```

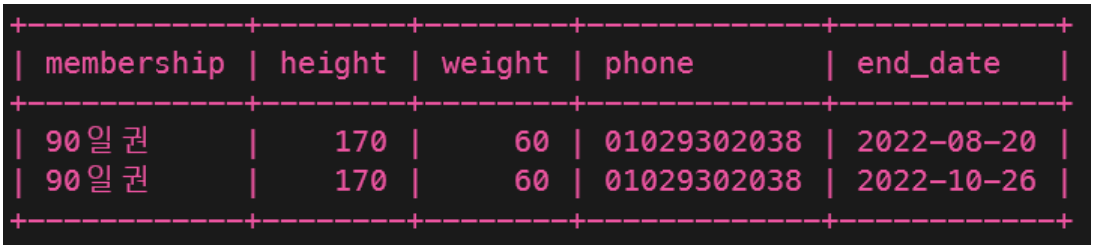
예시1. year(enroll_date) = 2022, month(enroll_date) = 5

branch	revenue
노량진	100000
수지	120000

```
-- MyPageFrame
select membership, height, weight, phone, end_date
from DB2022_members join DB2022_period using(member_id)
where DB2022_members.name=? and phone=? and password=?;
```

→ 입력된 사용자 정보(name, phone, password)에 해당하는 회원의 키, 몸무게, 전화번호와 회원권의 이름, 만료 일자를 가져온다.

예시2. name = '이나현', phone = 01029302038, password = 2940



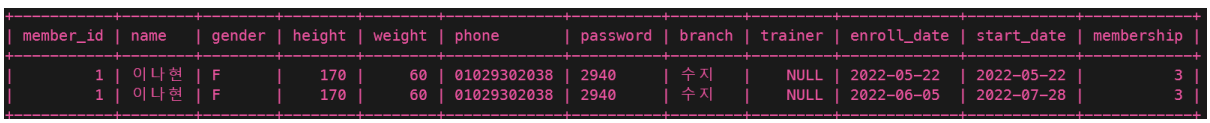
membership	height	weight	phone	end_date
90일권	170	60	01029302038	2022-08-20
90일권	170	60	01029302038	2022-10-26

```
-- TrainerFrame
-- 1)
select * from DB2022_members join DB2022_enroll
using(member_id) where name=? and phone=?;

-- 2)
select * from DB2022_trainers join DB2022_career
using(trainer_id) where branch ='수지';\r\n");
```

→ 트레이너 정보(name, phone)을 입력하면 회원과 회원권 정보 함께 가져오기

예시3. name = '이나현', phone = 01029302038



member_id	name	gender	height	weight	phone	password	branch	trainer	enroll_date	start_date	membership
1	이나현	F	170	60	01029302038	2940	수지	NULL	2022-05-22	2022-05-22	3
1	이나현	F	170	60	01029302038	2940	수지	NULL	2022-06-05	2022-07-28	3

→ 지점을 입력하면 트레이너와 트레이너 경력 정보를 함께 가져온다.

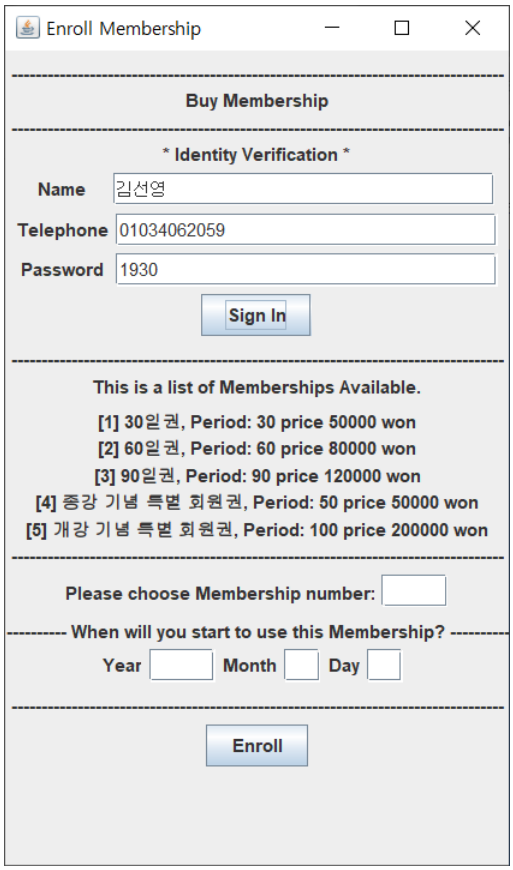
예시4. branch = '수지'

trainer_id	name	career_start_year	career_start_month	branch	password	career_year	career_month
2	정완섭	2022	6	수지	5860	0	0

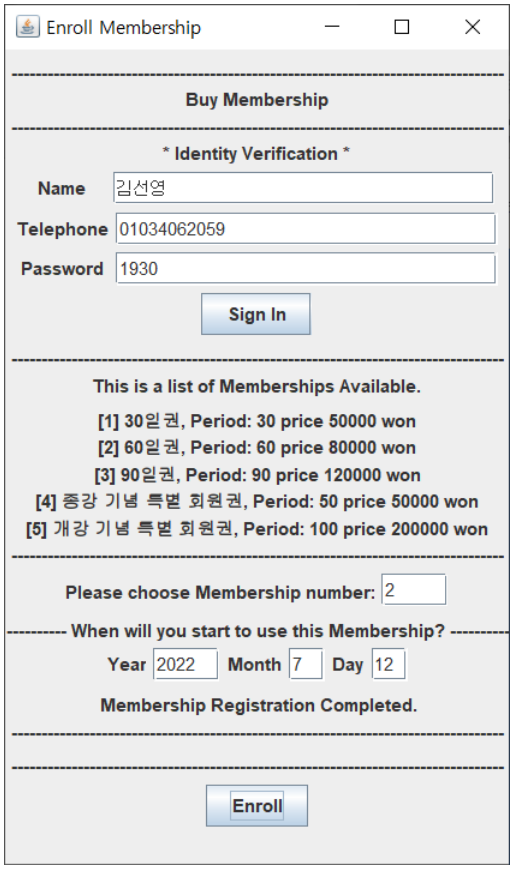
12. 매개 변수를 가지면서 동적으로 만드는 쿼리를 포함해야 한다. 다시 말해, 사용자로부터 입력 값을 받고 사용자가 입력한 값으로 쿼리를 생성한다.

회원권 등록 기능

회원권 등록 페이지에 접속하면 다음과 같은 화면이 출력된다.

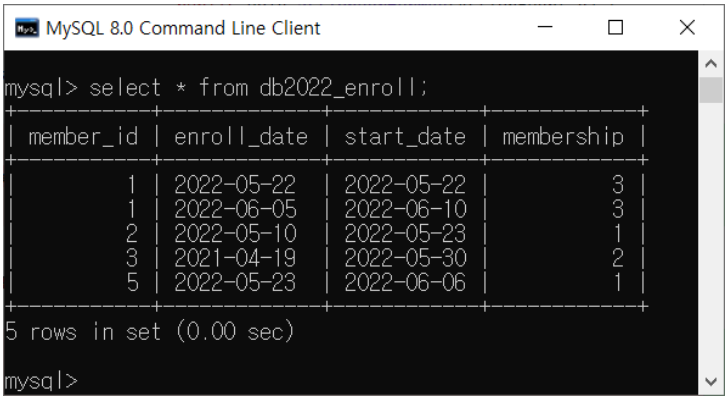


- 사용자는 본인 인증 후에 회원권 등록 서비스를 이용할 수 있으며 본인 인증을 위해서는 자신의 이름, 전화번호, 비밀번호를 정확하게 입력하여야 한다.
- 회원권의 종류와 해당 회원권의 번호를 상단에 출력하여 사용자가 보다 편리하게 회원권을 등록할 수 있게 하였다.

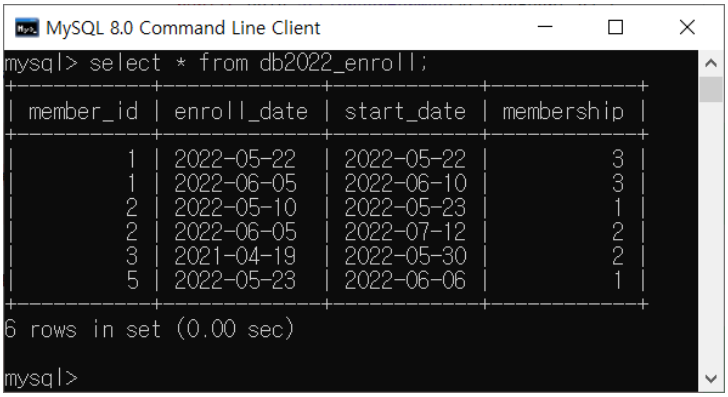


- 사용자가 회원권을 등록할 때에는 등록하고자 하는 회원권의 번호를 입력하고 해당 회원권을 사용할 날짜를 연도, 월, 일까지 정확하게 입력해야 한다.
- 회원권 테이블의 회원권 번호에는 사용자가 입력한 회원권 번호가, 시작 날짜에는 사용자가 입력한 시작 예정 날짜가 삽입되며 등록 날짜에는 회원권을 등록하는 날이 자동으로 삽입된다.

회원권 등록 전



회원권 등록 후



- 2번 회원이 등록한 2번 회원권이 정상적으로 추가된 것을 확인할 수 있다.
- 등록 날짜는 오늘(2022-06-05)로 자동 삽입되었고 시작 날짜는 회원이 입력한 날짜(2022-07-12)로 설정되어 있다.

코드

- 사용자가 본인 인증 시 입력했던 정보를 이용하여 해당 사용자를 선택하고 정보를 추출한다.

```
PreparedStatement pStmt = conn.prepareStatement("select * from DB2022_members where name=? and phone=?;");
pStmt.setString(1, member_name);
pStmt.setString(2, phone);
ResultSet info_set = pStmt.executeQuery();
```

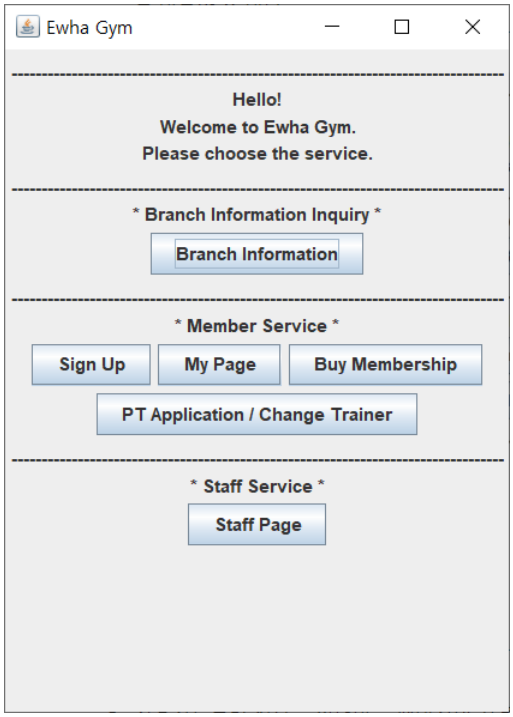
- 추출한 사용자 정보로부터 회원권 등록 시 넣어줘야 하는 회원 아이디를 얻을 수 있다.
- 그 외 시작 날짜와 회원권 아이디는 사용자가 입력한 값을 변수로 받아와 사용자가 입력한 값으로 삽입 쿼리를 생성한다.
- 등록 날짜는 LocalDate.now()를 이용하여 현재 회원권을 등록하고 있는 날짜를 삽입한다.

```
PreparedStatement pStmt = conn.prepareStatement("insert into db2022_enroll values(?,?,?,?);");
pStmt.setString(1, member_id);
pStmt.setDate(2, today);
pStmt.setDate(3, start_date);
pStmt.setString(4, membership_id);
pStmt.executeUpdate();
```

13. 그래픽 또는 문자 기반의 사용자 인터페이스를 사용해야 한다. 사용하기 쉽고 사용하기에 도움이 되는 정보를 가지고 있는 메뉴를 제공해야 한다.

프로그램을 처음 시작하면 다음과 같은 화면이 출력된다.

- 사용자의 편의를 위해 기능을 카테고리 별로 분류하였으며 사용자는 해당 기능 버튼을 클릭하여 상세 페이지로 이동할 수 있다.



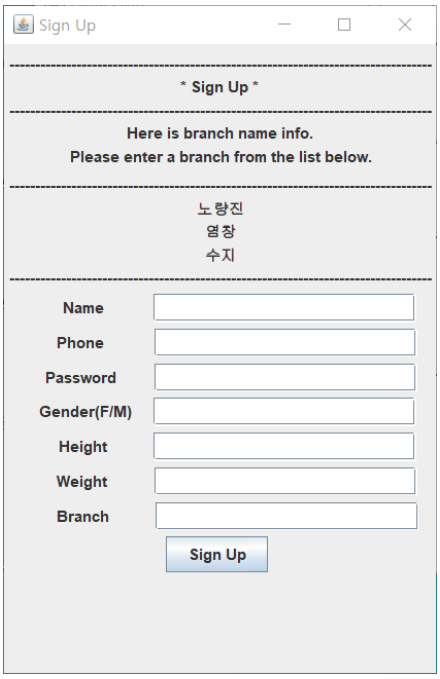
- 사용자로부터 정보를 입력받는 페이지의 경우 사용자가 정확한 정보를 입력할 수 있도록 조건에 어긋나는 데이터 입력 시 오류 메세지를 출력한다.
- 사용자는 해당 메세지를 참고하여 수정해야 할 정보의 위치를 빠르게 파악할 수 있다.

14. 데이터베이스에 삽입(insert)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다

회원가입

사용자가 회원 가입 버튼을 클릭하면 다음과 같은 화면이 출력된다.

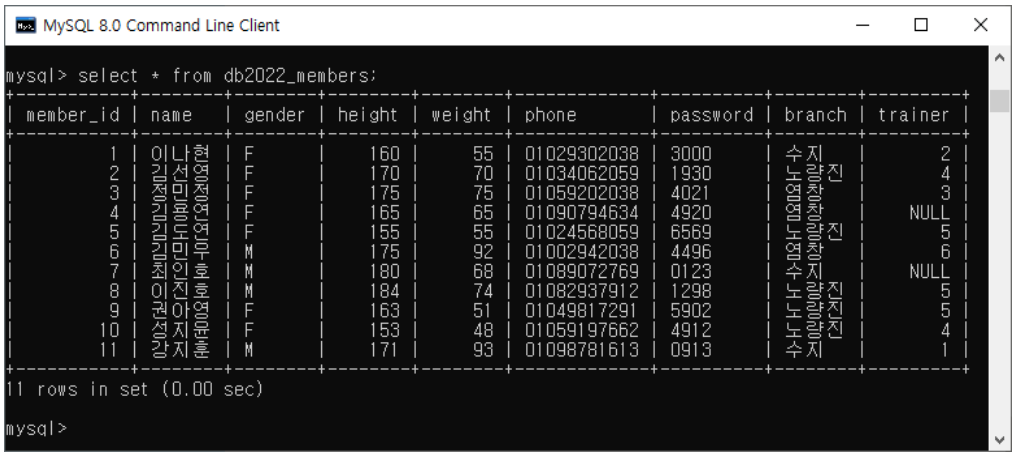
회원가입 전

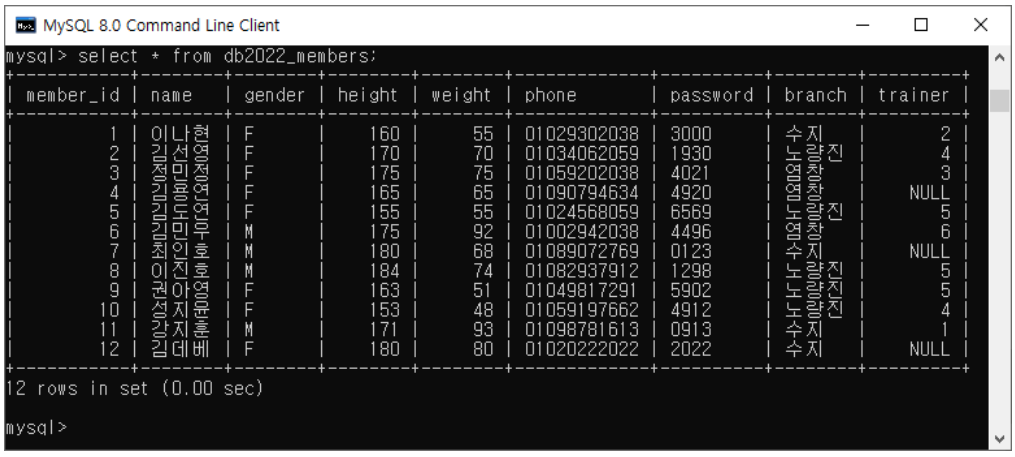


- 회원가입을 위해 필요한 정보 필드가 있고 사용자는 각 필드의 조건에 맞게 정보를 입력하여야 정상적으로 회원가입을 진행할 수 있다.
- 상단에 헬스장 지점 리스트를 출력하여 사용자로 하여금 더욱 편리하게 등록할 지점의 정보를 입력할 수 있게 하였다.

회원가입 후

- 모든 정보를 조건에 맞게 정확하게 입력하면 회원가입에 성공했다는 메세지와 함께 해당 회원의 정보다 데이터베이스에 반영된다.
- 12번에 해당 회원 정보가 추가된 것을 확인할 수 있다.





- 회원가입 시 자동으로 member_id가 부여되도록 코드를 작성하였기 때문에 12번에 해당 회원의 정보가 추가된 것을 확인할 수 있다.
- 최초 회원가입 시 담당 트레이너의 정보는 NULL로 저장되며 PT 등록을 원하는 회원은 PT 등록 페이지에 들어가 직접 트레이너의 정보를 확인하고 PT를 등록할 수 있다.

코드

- 회원가입 시 입력받은 정보를 이용하여 회원 테이블에 데이터를 삽입하는 쿼리를 생성한다.
- 추가적으로 회원 데이터를 추가할때 해당 데이터에 부여할 member_id를 구하기 위해 회원정보 테이블의 모든 튜플 개수를 세고 그 정보를 ResultSet에 저장하는 쿼리도 함께 생성한다.
- 사용자로부터 입력받은 정보와 직접 만든 member_id 값을 삽입 쿼리에 이용하여 회원 테이블에 데이터를 추가할 수 있다.

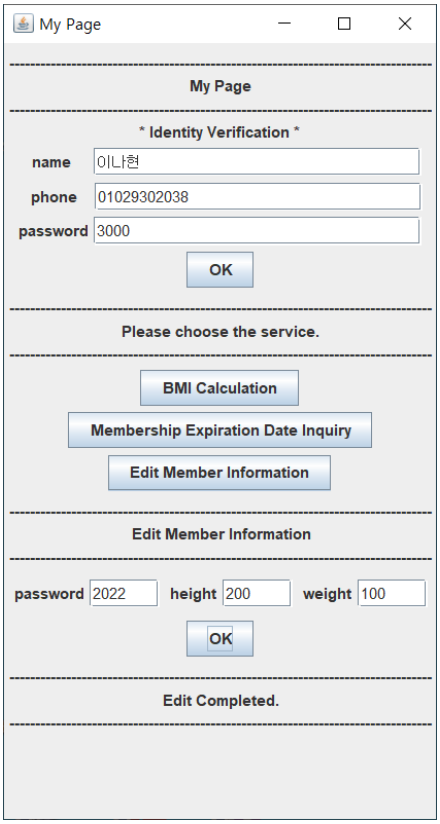
```
// 회원가입 시 사용자 정보 insert하는 쿼리
PreparedStatement pStmt = conn.prepareStatement("insert into DB2022_members values(?,?,?,?,?,?,?,?,?)");
// member_id pk값 자동부여하기 위해 데이터 개수 count하는 쿼리
ResultSet rSet = stmt.executeQuery("select count(*) as num from DB2022_members;");

if (rSet.next()) {
    int n = rSet.getInt("num") + 1; // 총 회원 수에 1을 더하여 member_id로 지정한다.
    System.out.println(n);
    pStmt.setInt(1, n);
    pStmt.setString(2, name);
    pStmt.setString(3, gender);
    pStmt.setInt(4, Integer.parseInt(height));
    pStmt.setInt(5, Integer.parseInt(weight));
    pStmt.setString(6, phone);
    pStmt.setString(7, passwd);
    pStmt.setString(8, branch);
    pStmt.setString(9, null);
    pStmt.executeUpdate();
    // 회원가입 완료 메시지 띄우기
    label.setText("<html><body style='text-align:center;'>"
        + "-----------------------------------------------------------------------------------<br/>"
        + "Your registration is completed<br/>"
        + "-----------------------------------------------------------------------------------<br/>"
        + "</body></html>");
```

15. 데이터베이스에 갱신(update)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.

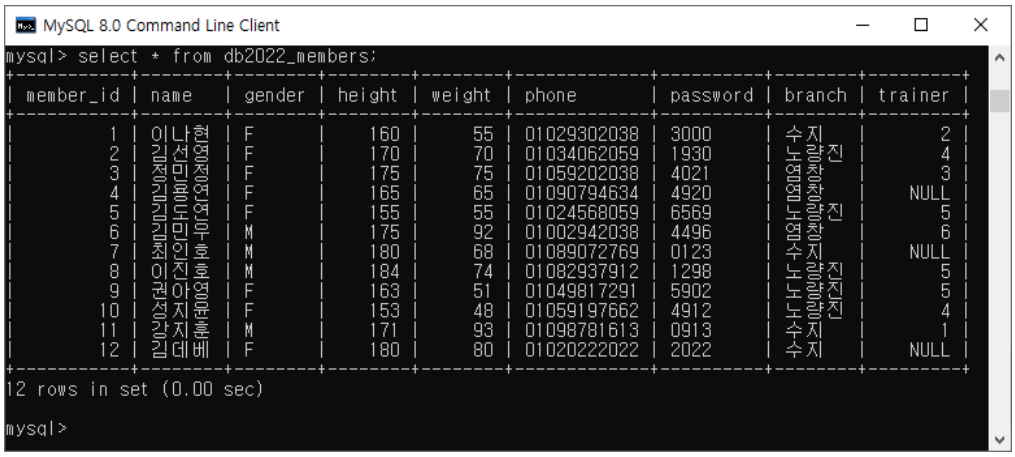
회원정보 수정 기능

회원정보 수정을 위해 마이페이지에 접속하면 다음과 같은 화면이 출력된다.



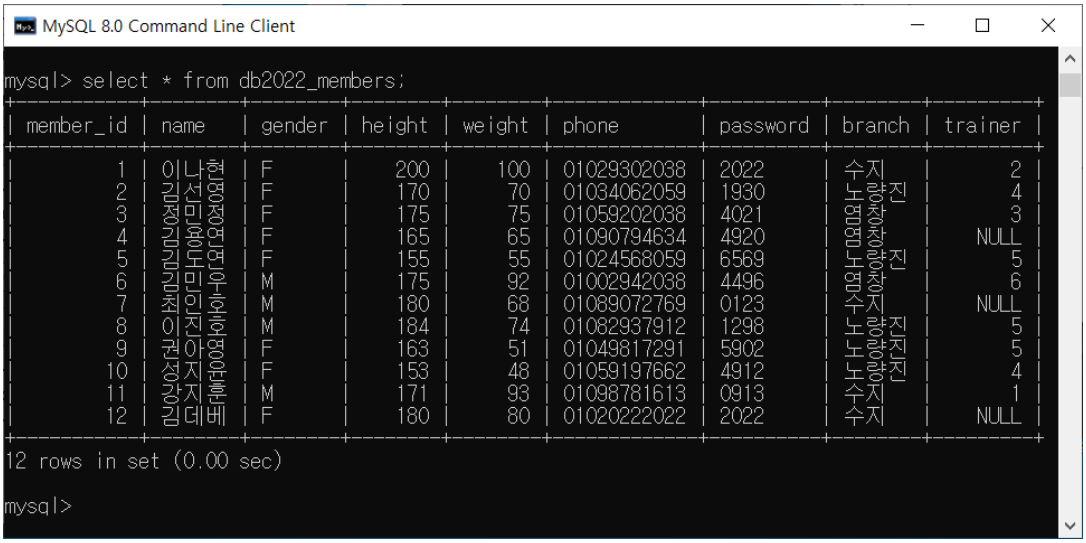
- 사용자는 비밀번호, 키, 몸무게 정보를 수정할 수 있다.
- 4자리 비밀번호 조건에 맞게 입력란에 정보를 입력하여야 수정한 정보를 데이터베이스에 반영할 수 있다.

회원정보 수정 전



1번 회원 회원정보 수정 후

- 비밀번호 3000 → 2022
- 키 160 → 200
- 몸무게 55 → 100



1번 회원의 키, 몸무게, 비밀번호 정보가 수정된 것을 확인할 수 있다.

코드

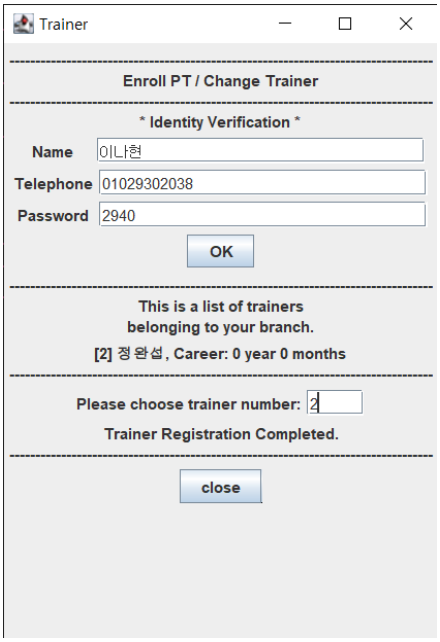
- 회원 정보 수정 페이지에 사용자가 입력한 정보를 변수로 받아온다.

```
String new_passwd = edit_passwd.getText().trim();
String new_height = edit_height.getText().trim();
String new_weight = edit_weight.getText().trim();
```

- 변수로 받아온 정보를 이용하여 회원 정보 갱신 쿼리를 생성한다..

```
PreparedStatement pStmt = conn.prepareStatement("update DB2022_members set password=?, height=?, weight=? where
 phone=?;");
pStmt.setString(1, new_passwd);
pStmt.setString(2, new_height);
pStmt.setString(3, new_weight);
pStmt.setString(4, phone);
pStmt.executeUpdate();
```

트레이너 등록/변경



- 사용자는 본인 인증을 하면 트레이너 목록을 볼 수 있다.
- 원하는 트레이너 번호를 입력하면 PT 신청/트레이너 변경이 완료된다.

트레이너 변경 전

member_id	name	gender	height	weight	phone	password	branch	trainer
1	이나현	F	170	60	01029302038	2940	수지	NULL

PT를 신청하지 않아 trainer가 NULL 이다.

트레이너 변경 후

member_id	name	gender	height	weight	phone	password	branch	trainer
1	이나현	F	170	60	01029302038	2940	수지	2

트레이너를 선택 후 trainer 정보가 변경된 것을 볼 수 있다.

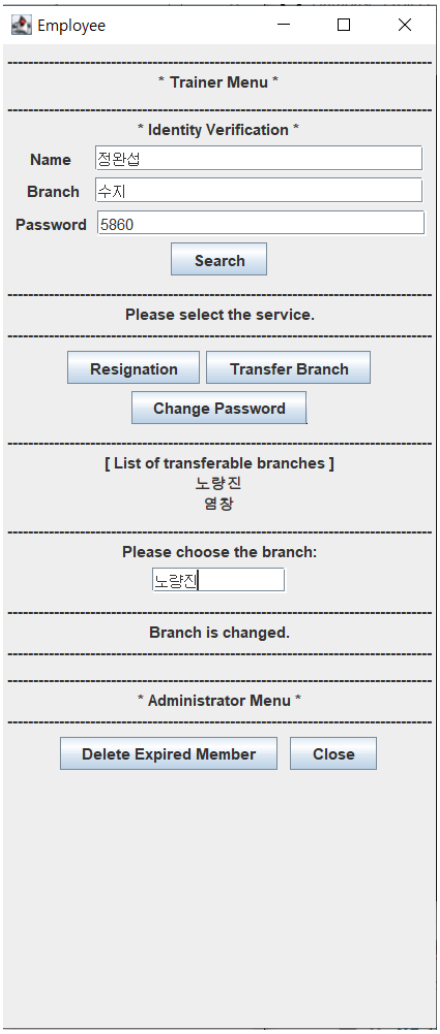
코드

- 해당 회원의 trainer 정보를 update 한다.

```
PreparedStatement pStmt = conn.prepareStatement("update DB2022_members set trainer =? where member_id=?;");
pStmt.setString(1, trainer_id);
pStmt.setString(2, member_id);
```

트레이너 지점 이동

- 트레이너가 본인 인증을 하면 원하는 지점으로 이동할 수 있는 GUI가 생성된다.



트레이너 지점 변경 전

trainer_id	name	career_start_year	career_start_month	branch	password
2	정완섭	2022	6	수지	5860

지점(branch) 변경 전 트레이너 지점이 '수지'이다.

트레이너 지점 변경 후

trainer_id	name	career_start_year	career_start_month	branch	password
2	정완섭	2022	6	노량진	5860

지점(branch) 변경 후 트레이너 지점이 '노량진'이다.

코드

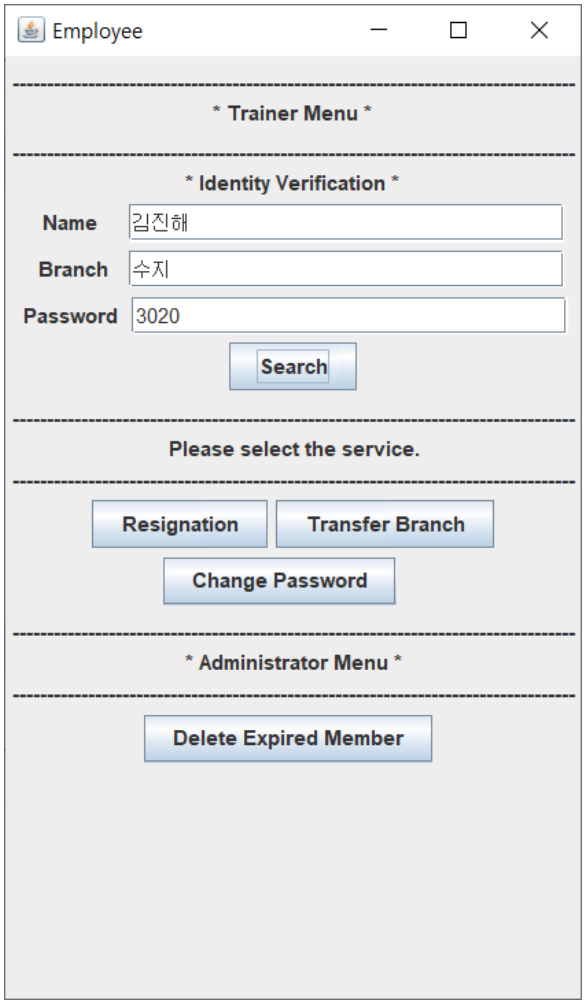
```
-- 1) 이동된 트레이너가 전 지점에서 맡은 회원의 trainer를 NULL로 설정한다.
PreparedStatement pStmt = conn.prepareStatement("update DB2022_members set trainer = null where trainer=?;");
pStmt.setString(1, trainer);
pStmt.executeUpdate();

-- 2) 입력 받은 지점 변수로 트레이너 정보를 업데이트 한다.
```

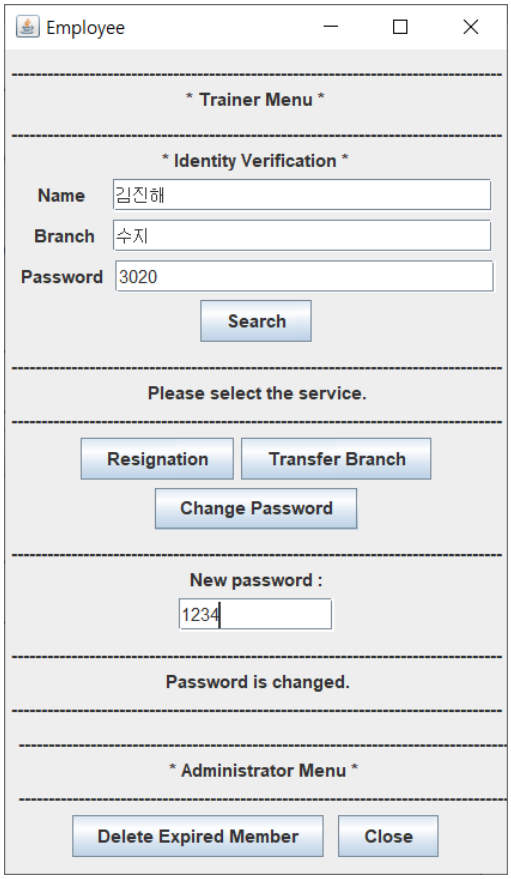
```
pStmt = conn.prepareStatement("update DB2022_trainers set branch =? where trainer_id=?;");
pStmt.setString(1, branch);
pStmt.setString(2, trainer);
pStmt.executeUpdate();
```

트레이너 비밀번호 변경

트레이너가 트레이너 페이지에 접속하면 다음과 같은 화면이 출력된다.



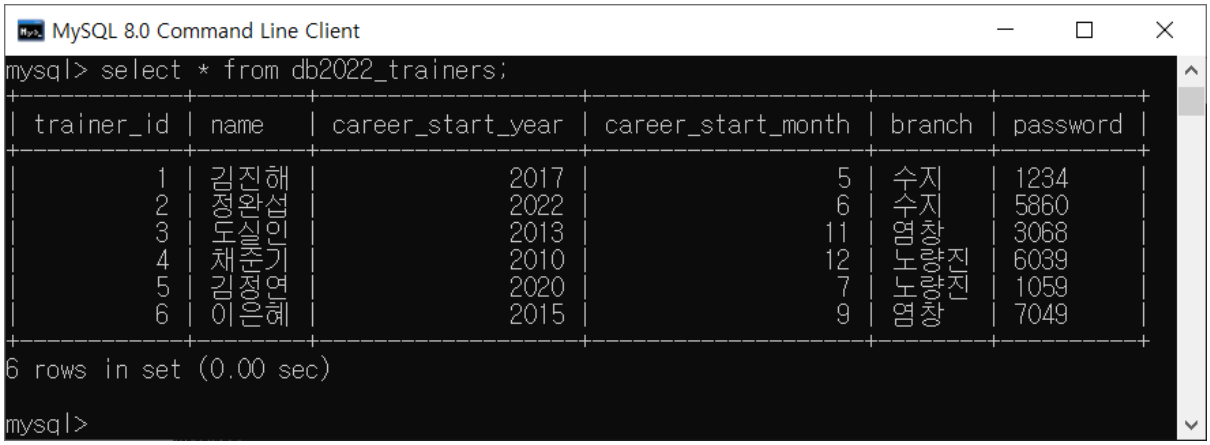
- 본인의 이름과 지점 정보, 비밀번호를 입력하면 아래와 같이 트레이너 서비스를 이용할 수 있는 버튼이 출력된다.



- 비밀번호 변경 버튼 클릭 후 변경할 비밀번호를 입력하면 본인의 비밀번호를 변경할 수 있다.

비밀번호 변경 전

비밀번호 변경 후



- 트레이너가 입력한 새로운 비밀번호로 변경된 것을 확인할 수 있다.

코드

```
PreparedStatement pStmt = conn.prepareStatement(
             "select trainer_id, branch from DB2022_trainers where name=? and branch=? and password=?;");
pStmt.setString(1, name);
pStmt.setString(2, branch);
pStmt.setString(3, passwd);
ResultSet rset = pStmt.executeQuery();
```

- 처음 트레이너가 본인 인증을 할때 입력한 정보를 이용하여 해당 트레이너의 정보를 가져오는 검색 쿼리를 생성한다.

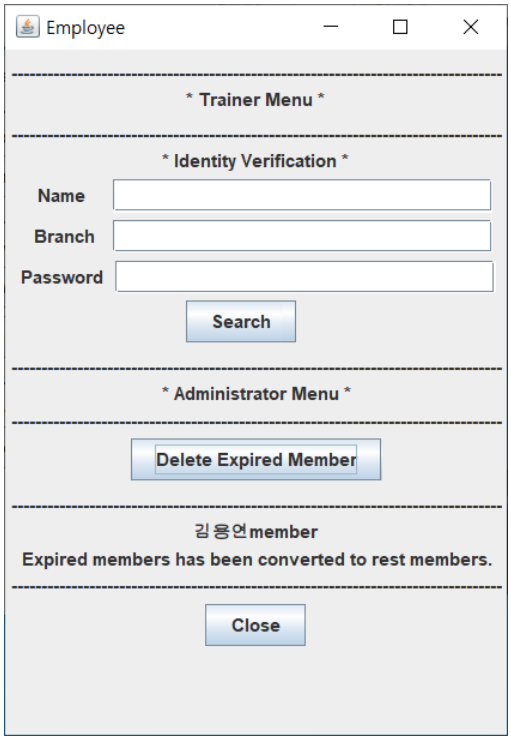
```
PreparedStatement pStmt = conn
          .prepareStatement("" + "update DB2022_trainers set password=? where trainer_id=?");
pStmt.setString(1, pwd);
pStmt.setString(2, trainer_id);
pStmt.executeUpdate();
```

- 검색 쿼리를 이용하여 가져온 트레이너 정보 중 트레이너 아이디를 이용하여 해당 아이디에 해당하는 트레이너의 정보를 갱신하는 쿼리를 생성한다.
- 트레이너의 비밀번호를 갱신할 때에는 사용자가 입력한 비밀번호를 변수로 받아와 쿼리에 적용한다.

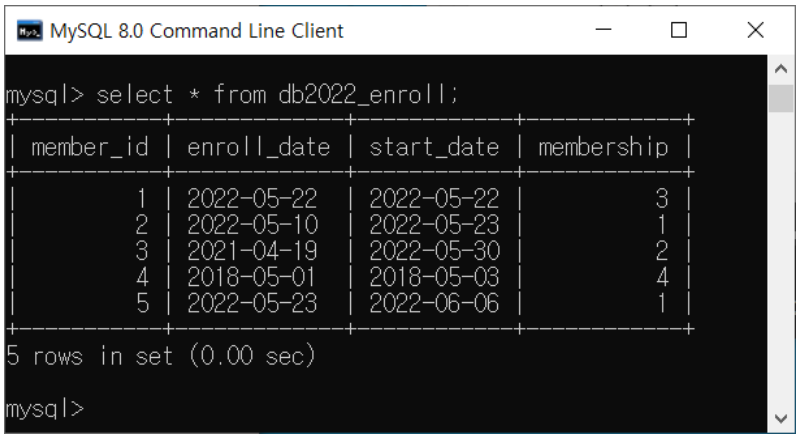
16. 데이터베이스에 삭제(delete)를 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다

회원권 만료 회원 삭제 기능

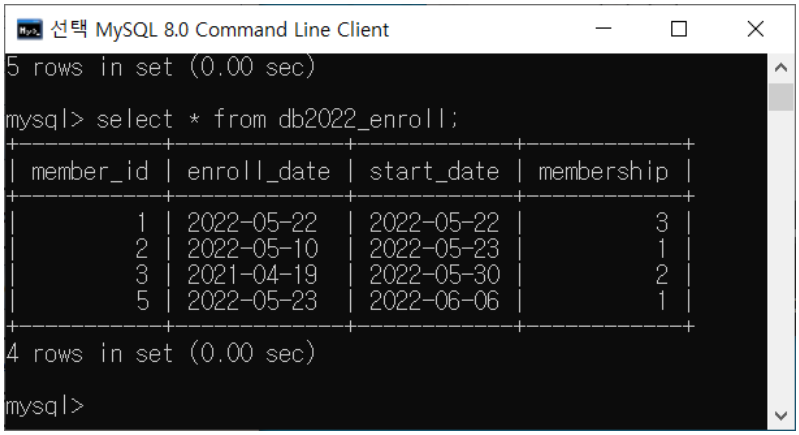
직원 페이지에 접속하면 다음과 같은 화면이 출력된다.



- 헬스장 직원은 만료 회원 삭제 버튼을 클릭하여 해당 기능을 사용할 수 있다.
- 만료 회원 삭제 버튼 클릭 시 회원권이 만료된 회원의 리스트가 출력되고 해당 회원은 enroll 테이블에서 자동으로 삭제된다.



버튼 클릭 전



버튼 클릭 후

코드

- 오늘 날짜 기준 회원권이 만료된 회원 리스트를 선택하여 ResultSet에 저장한다.

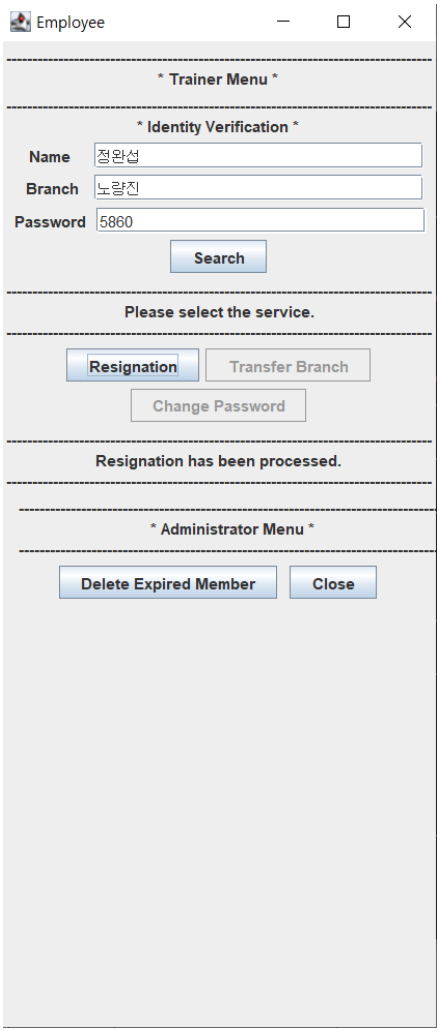
```
ResultSet rset = stmt.executeQuery("select name,member_id from DB2022_period where timestampdiff(day,end_date,curdat
e())>0;");
```

- member_id에 해당하는 데이터 튜플을 삭제하는 쿼리를 생성하고 ResultSet에 저장된 만료 회원 리스트 정보 중 member_id를 삭제 쿼리에 이용한다.

```
PreparedStatement pStmt;
pStmt = conn.prepareStatement("delete from DB2022_enroll where member_id=?");
pStmt.setString(1, rset.getString("member_id"));
pStmt.executeUpdate();
```

퇴사 트레이너 삭제 기능

본인 인증 후 Staff Page에 접속하면 다음과 같은 화면이 뜬다.



- Resignation 퇴사 버튼을 누르면 DB에서 트레이너 정보를 삭제한다.

trainer_id	name	career_start_year	career_start_month	branch	password
2	정완섭	2022	6	노량진	5860
3	도실인	2013	11	염창	3068
4	채준기	2010	12	노량진	6039
5	김정연	2020	7	노량진	1059
6	이은혜	2015	9	염창	7049

'정완섭' 트레이너 퇴사 전

trainer_id	name	career_start_year	career_start_month	branch	password
3	도실인	2013	11	염창	3068
4	채준기	2010	12	노량진	6039
5	김정연	2020	7	노량진	1059
6	이은혜	2015	9	염창	7049

'정완섭' 트레이너 퇴사 후 삭제

코드

- 입력 받은 trainer_id로 해당 트레이너를 삭제한다.

```
// resign the trainer
pStmt = conn.prepareStatement("delete from DB2022_trainers where trainer_id=?;");
pStmt.setString(1, trainer);
pStmt.executeUpdate();
```

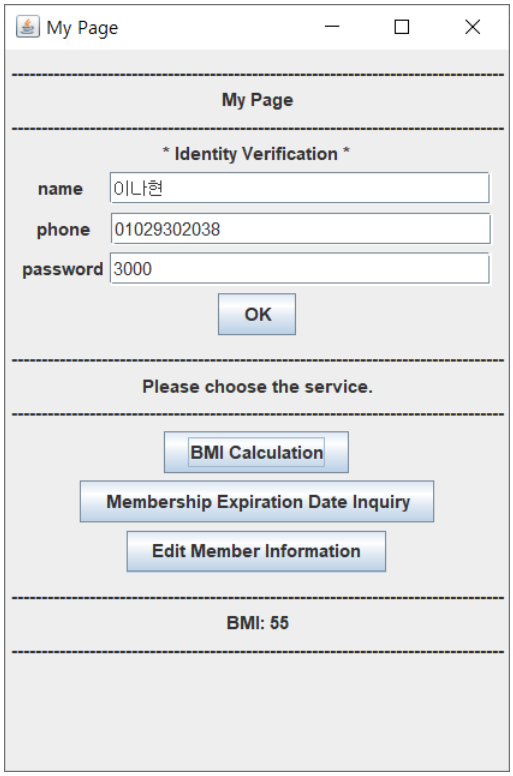
17. 데이터베이스에 검색(select)을 하기 위한 인터페이스와 쿼리를 가지고 있어야 한다.

BMI 계산 기능

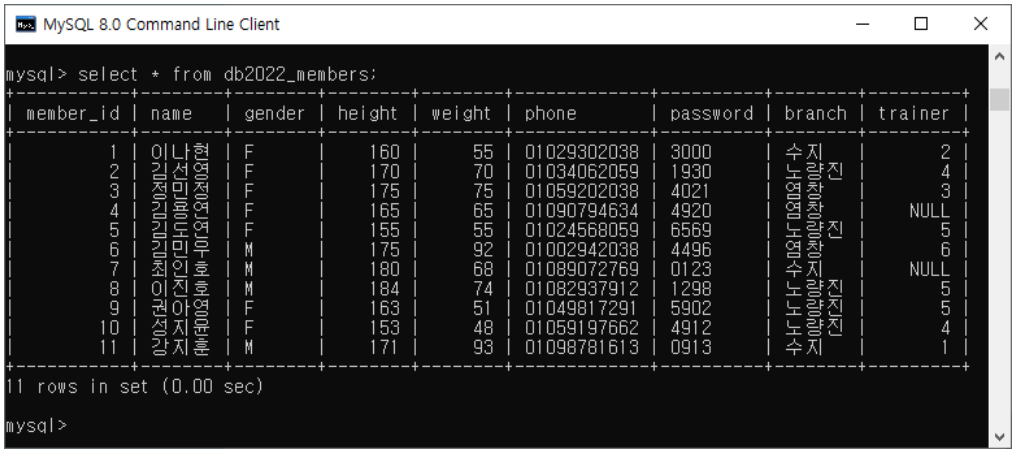
사용자로부터 입력 받은 정보 바탕으로 사용자를 검색하여 BMI 점수 계산하여 출력하는 기능

```
PreparedStatement pStmt = conn.prepareStatement(
                    "select height, weight, phone, end_date from DB2022_members join DB2022_period where DB2022_mem
bers.name=? and phone=? and password=?;");
pStmt.setString(1, name);
pStmt.setString(2, phone);
pStmt.setString(3, passwd);
ResultSet rset = pStmt.executeQuery();
```

마이페이지에 접속하면 다음과 같은 화면이 출력된다.



- 해당 화면에서 BMI 계산 버튼을 클릭하면 사용자가 가입 시 입력하였던 키, 몸무게 정보를 바탕으로 계산된 BMI 점수를 확인할 수 있다.
- 사용자가 마이페이지 접속 시 입력한 이름, 전화번호, 비밀번호를 이용하여 회원 정보 테이블에서 해당 사용자의 회원 정보를 찾고 그 정보를 바탕으로 BMI를 계산한다.



코드

- 사용자로부터 입력받은 정보를 이용하여 회원을 선택하는 쿼리를 생성한다.

where 절에서 이름, 전화번호, 비밀번호가 모두 일치하는지 확인 후 회원을 선택한다.

```
PreparedStatement pStmt = conn.prepareStatement("select height, weight, phone, end_date from DB2022_members join DB2
022_period where DB2022_members.name=? and phone=? and password=?;");
pStmt.setString(1, name);
pStmt.setString(2, phone);
pStmt.setString(3, passwd);
ResultSet rset = pStmt.executeQuery();
```

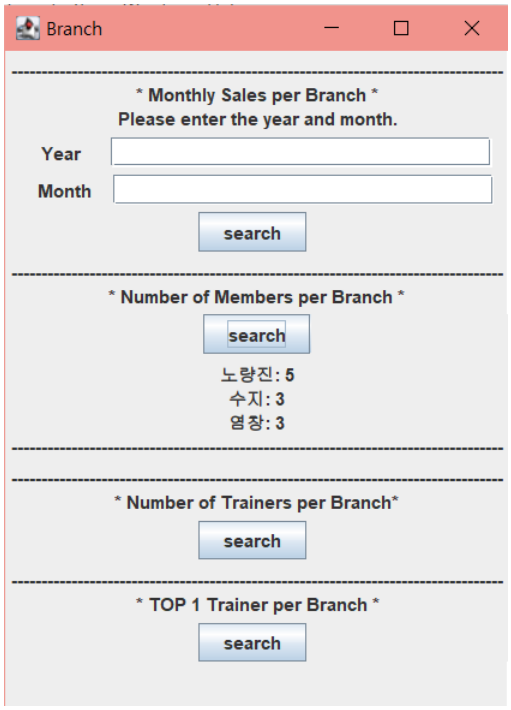
- 선택한 사용자로부터 추출한 키, 몸무게 값 바탕으로 BMI 점수 계산하여 출력

```
int height_num = Integer.parseInt(height);
int weight_num = Integer.parseInt(weight);

int BMI = weight_num/((height_num/100)*(height_num/100));
```

지점 별 누적 회원 수 보여주기 기능

버튼을 누르면 지점별 누적 회원 수를 보여준다.

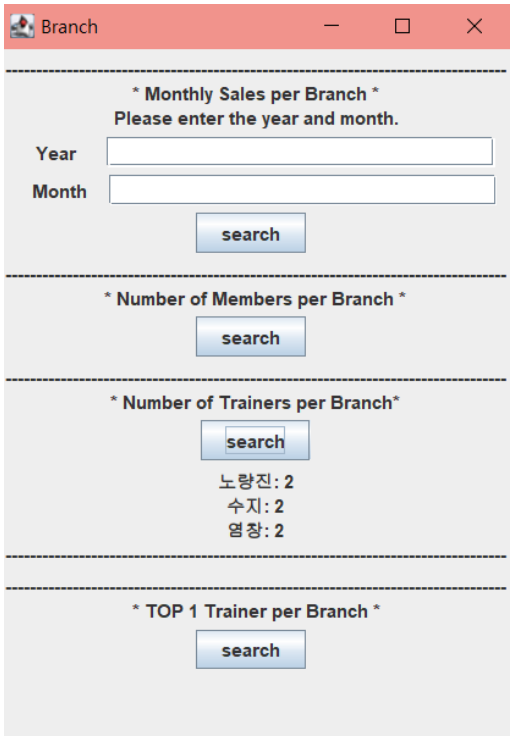



```
// 버튼에 지점별 회원 수 보여주는 이벤트 추가
      member_btn.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            try(Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBI
D,
                  DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();){
               ResultSet rset = stmt
                           .executeQuery("select branch, count(*) from DB2022_members group by branch;");
               member_num.setText("<html><body style='text-align:center;'>");
                     while (rset.next()) {
                        String str = (rset.getString("branch") + ": " + rset.getInt(2)+"<br/>");
                        member_num.setText(member_num.getText()+str);
                     }
                     member_num.setText(member_num.getText()
                           + "-----------------------------------------------------------------------------------------<br/
></body></html>");
            }catch(SQLException sqle) {
               System.out.println("SQL Exception: "+sqle);
            }
         }

      });
```

지점 별 트레이너 수 보여주기 기능

버튼을 누르면 지점별 트레이너 수를 보여준다.



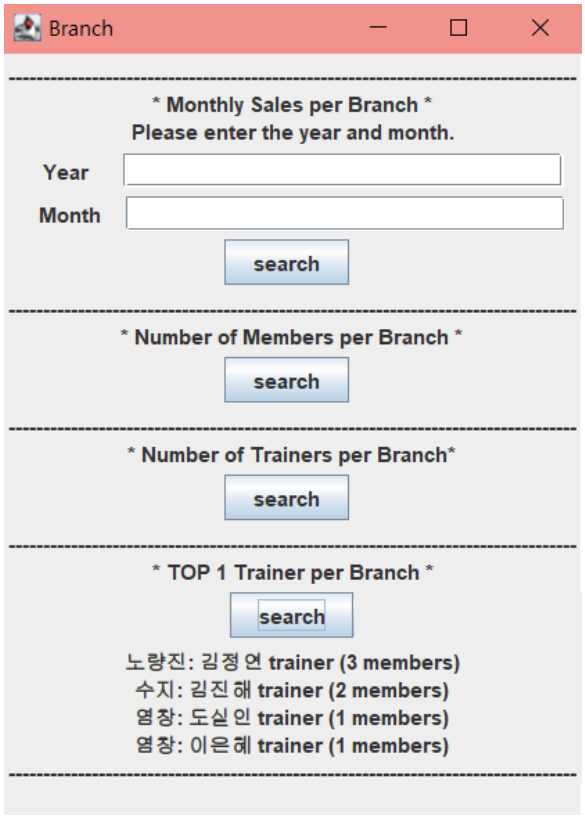
코드

```
// 버튼에 지점별 트레이너 수 보여주는 이벤트 추가
      trainer_btn.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            try(Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DBI
D,
                  DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();){
               ResultSet rset = stmt
                           .executeQuery("select branch, count(*) from DB2022_trainers group by branch;");
               trainer_num.setText("<html><body style='text-align:center;'>");
                     while (rset.next()) {
                        String str = (rset.getString("branch") + ": " + rset.getInt(2)+"<br/>");
                        trainer_num.setText(trainer_num.getText()+str);
                     }
                     trainer_num.setText(trainer_num.getText()
                           + "-------------------------------------------------------------------------------------<br/
>"
                           + "</body></html>");
            }catch(SQLException sqle) {
               System.out.println("SQL Exception: "+sqle);
            }
         }

      });
```

지점 별 누적 회원 수 1위 트레이너 보여주기 기능

버튼을 누르면 지점별 누적 회원 수 1위 트레이너를 보여준다.

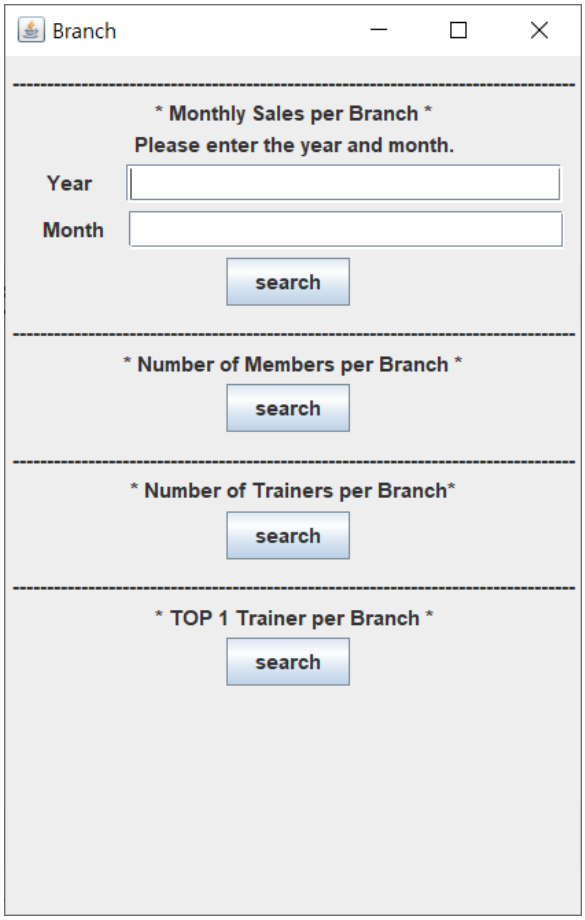


코드

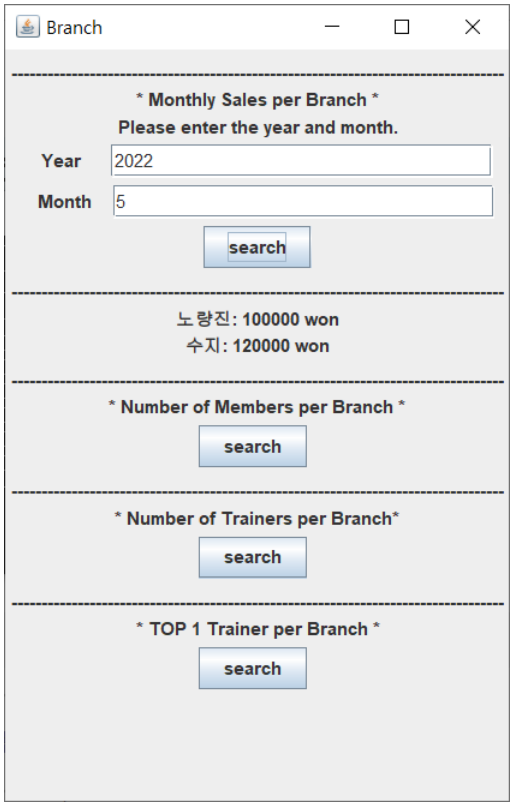
```
// 버튼에 지점별 회원 수 1위 트레이너 보여주는 이벤트 추가
      top1_btn.addActionListener(new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
             try(Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/" + DB2022Team07_main.DB
ID,
                   DB2022Team07_main.USERID, DB2022Team07_main.PASSWD); Statement stmt = conn.createStatement();){
                ResultSet rset = stmt
                            .executeQuery("with A as (select trainer, DB2022_trainers.branch, DB2022_trainers.name,
 count(trainer) as totalMemb from DB2022_trainers, DB2022_members "
                                  + "where DB2022_members.branch = DB2022_trainers.branch and DB2022_members.trainer
= DB2022_trainers.trainer_id group by trainer), \r\n"
                                  + "B as (select branch, max(totalMemb) as topMemb from A group by branch) \r\n"
                                  + "select A.branch, name, topMemb from A,B where A.branch = B.branch and totalMemb
= topMemb order by branch; \r\n"
                                  + "\r\n"
                                  + "");
                top1_label.setText("<html><body style='text-align:center;'>");
                      while (rset.next()) {
                         String str = (rset.getString("branch") + ": " + rset.getString("name")+ " trainer (" + rse
t.getInt(3)+ " members)" );
                         top1_label.setText(top1_label.getText() + str+"<br/>");
                         System.out.println(str);
                      }
                      top1_label.setText(top1_label.getText()+"----------------------------------------------------
----------------------------<br/></body></html>");
             } catch(SQLException sqle) {
                System.out.println("SQL Exception: "+sqle);
             }
          }
       });
```

지점 별 월 매출 계산

사용자가 지점 페이지에 접속하면 다음과 같은 화면이 출력된다.



- 사용자는 매출을 조회하고 싶은 연도와 월을 입력한다.
- 매출은 회원권 등록 날짜를 기준으로 계산된다.



코드

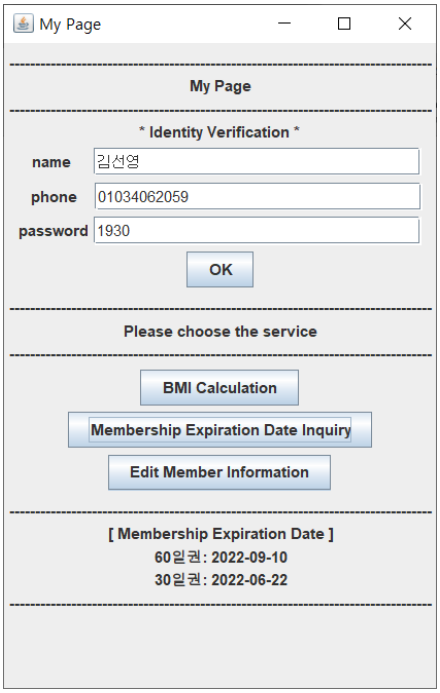
```
PreparedStatement pStmt = conn.prepareStatement(
                  "select branch, sum(price) as 'revenue' " + "from DB2022_enroll join DB2022_membership "
                        + "on DB2022_enroll.membership=DB2022_membership.membership_id "
                        + "join DB2022_members using(member_id) "
                        + "where year(enroll_date)=? and month(enroll_date)=? " + "group by branch "
                        + "order by branch;");
pStmt.setString(1, tempYear);
pStmt.setString(2, tempMonth);
ResultSet rset = pStmt.executeQuery();
```

- 사용자로부터 연도와 월을 변수로 받아와 검색 쿼리를 생성한다.
- 데이터를 지점별로 그룹화하여 지점별 매출을 구한다.

회원권 만료 회원 조회 기능

회원권 만료 날짜 조회

사용자가 마이페이지에 접속하면 다음과 같은 페이지가 출력된다.



- 회원권 만료 날짜 조회 버튼을 클릭하면 본인의 회원권 만료 날짜를 확인할 수 있다.
- 회원권 만료 날짜는 등록 날짜가 아닌 시작 날짜를 기준으로 계산된다.
- 보유하고 있는 회원권이 여러개인 경우 각각의 만료 날짜가 모두 출력된다.

코드

```
PreparedStatement pStmt = conn.prepareStatement(
              "select membership, height, weight, phone, end_date "
              + "from DB2022_members join DB2022_period using(member_id) "
              + "where DB2022_members.name=? and phone=? and password=?;");
pStmt.setString(1, name);
pStmt.setString(2, phone);
pStmt.setString(3, passwd);
ResultSet rset = pStmt.executeQuery();
```

- 사용자가 마이페이지에 접속하여 본인 인증을 할 때 입력한 정보를 이용하여 해당 회원의 정보를 가져오는 검색 쿼리를 생성한다.
- 회원 테이블과 회원권 기간 view를 join하여 사용하며 view의 end_date를 가져와 만료 날짜를 출력할 때 이용한다.

7. 요구 조건 외의 팀만의 강점

- 사용자 입력의 다양한 경우의 수를 고려하여 조건에 맞지 않은 입력이 주어졌을 때의 분기 처리를 상세히 작성하여 데이터베이스에 잘못된 정보가 저장되지 않도록 구현하였다. 분기 처리를 통해 예방한 상황의 예시는 아래와 같다.
  - 잘못된 형식의 비밀번호로 변경
  - 존재하지 않는 지점으로 트레이너 지점 이동 신청
  - 회원권 사용 시작 날짜를 회원권 등록 날짜보다 이른 날짜로 등록
  - 지점에 소속되지 않은 트레이너로 등록/변경
- 사용자의 입력을 돕는 유저 친화적인 인터페이스 제공하였다. 유저 친화적인 인터페이스의 예시는 아래와 같다.
  - 회원권 등록 과정에서 회원권 등록 날짜는 오늘 날짜로 자동 입력되도록 구현

- 회원 가입 과정에서 지점 이름 목록을 보고 쓸 수 있도록 구성
- 회원 가입 과정에서 회원의 id는 자동으로 생성되도록 구성
- 트레이너 등록/변경 과정에서 트레이너의 이름과 트레이너의 경력 정보를 함께 제공하여 회원이 트레이너를 선택하는데 도움이 되도록 구성
- 오늘 날짜와 회원권의 만료 날짜를 자동으로 비교하여 만료된 회원권을 한번에 삭제하도록 구현
- 회원의 본인 확인 단계에서 회원이 전화번호를 xxx-xxxx-xxxx형식으로 입력해도, 자동으로 올바른 형식으로 변환되어 입력되도록 구현
- 회원이 정보를 잘못 입력한 경우, 잘못 입력된 부분의 위치를 메세지로 알려주어 사용자가 수정하는데 도움이 되도록 구성

- 회원의 정보에 접근하거나 트레이너의 정보에 접근하는 서비스에서는 비밀번호를 입력 받는 본인 확인 단계를 진행하였다. 다른 사람이 다른 사용자의 데이터베이스 정보에 접근하지 못하도록 하여 개인 정보의 유출을 예방했다.

8. 팀의 구성원의 담당 부분

> 각 팀 구성원의 프로젝트 수행 부분.

김도연	- ER Diagram 및 Schema의 구조 brainstorming - DDL SQL 문 작성 - GUI 구현, 회원 가입, 회원권 만료 회원 리스트 출력 및 만료 회원 삭제, 회원별 BMI 점수 계산, 회원 정보 수정, 주석 작성 - 레포트 작성
김선영	- DDL SQL 문 작성 - 회원권 등록 기능 로직 개발 및 관련 쿼리 작성 - EnrollFrame 인터페이스 구현 - 레포트 작성 - 최종 발표
김용연	- DDL SQL 문 작성 - 회원 가입 GUI 구현 (수정됨) - 지점 별 Top 3 트레이너 구하는 기능 추가 (Java 코드가 많아서 수정됨) - 코드 주석 작성 - 레포트 작성
이나현	- ER Diagram 및 Schema의 구조 brainstorming - DDL SQL 문 작성 - 지점별 회원 수 출력, 지점별 트레이너 수 출력, 지점별 누적 회원수 Top1 트레이너 출력 - sql문 주석 작성 - 레포트 작성
정민정	- ER Diagram 및 Schema의 구조 brainstorming - 최종 ER Diagram 및 Relation Schema 제작 - DDL SQL 문 작성 - GUI 구현, 지점별 월 매출 계산, 트레이너 등록/변경, 트레이너 지점 이동, 트레이너 퇴사, 트레이너 비밀번호 변경, 회원권 만료 날짜 조회 - 레포트 작성