데이터베이스 미니프로젝트3 최종 보고서

<GYM Management System>

컴퓨터공학과

19011460

이유재

목차

I. 수정사항

1. Purchase Relation에서 Member, Goods, Data 간 Cardinality를 N:M:1에서 N:M:L로 수정
2. Program에서 number_of_member를 강의의 최대 정원과 현재 신청 인원으로 구분함
3. Purchase History에서 날짜 속성을 삭제
   - 회원이 해당 물품에 대한 물품 내역을 누적하도록 변경
4. 미니프로젝트2에서 member table, trainer table에 있는 phone_number 속성을 따로 빼내에 테이블로 만들었는데 그러지 않고 member, trainer table 내부로 다시 넣음
5. Program테이블에 program_no를 넣어 Primary Key로 설정
   - Participate 테이블에서 Program을 참조할 때 program_no를 사용함
6. Member Table에서 자기 참조시 참조 제약 조건 삭제

II. 요구사항 분석

1. 운동 센터의 모든 직원은 트레이너로 이루어짐.
2. 회원 등록할 때 다음과 같은 정보들을 요구.
   - 회원 번호, 이름, 연락처, 성별, 주소, 결제 금액, 추천인 회원 번호
   - 회원 식별은 회원 번호로 함.
3. 트레이너 관리를 위해 트레이너에 대해 다음과 같은 정보를 요구한다.
   - 트레이너 번호, 이름, 연락처, 성별, 주소, 급여
   - 트레이너 식별은 트레이너 번호로 함
4. 프로그램은 트레이너가 개설한 수업에 대해 다음과 같은 정보들을 요구
   - 프로그램 번호, 프로그램 명, 프로그램 타입, 프로그램 트레이너, 프로그램 가격, 현재 수강인원, 최대 수강인원
   - 프로그램 식별은 프로그램 번호로 함
   - 프로그램이 만들어지고 담당 트레이너가 부여될 수도 있다.
   - 프로그램의 담당 트레이너가 도중에 그만둘 경우 해당 프로그램의 담당은 우선적으로 트레이너의 번호가 가장 빠른 트레이너가 담당함

- 회원이 프로그램을 환불할 때 전액 환불을 받는다.

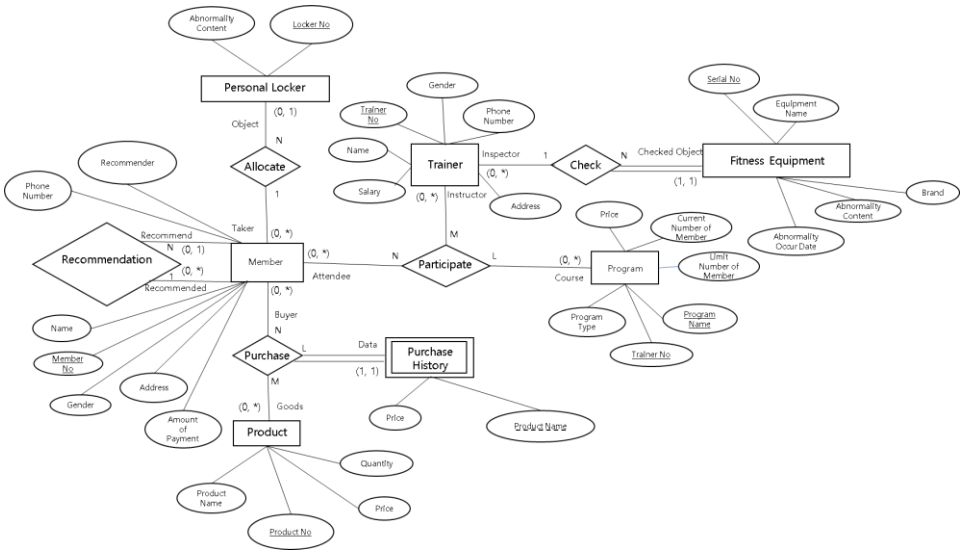
5. 트레이너는 운동기구를 점검 후 이상 유무를 기록해야 하며, 이를 위해 Fitness equipment는 다음과 같은 정보들을 요구함.
   - 기구 일련번호, 브랜드, 기구 이름, 이상 증세, 이상 발생일
   - 운동 기구 식별은 제품 일련 번호로 함
6. 운동 센터에서는 운동 용품, 음료 등 여러 제품을 판매하여, 이를 위해 Product entity는 다음과 같은 정보를 요구한다.
   - 제품 번호, 제품명, 가격, 수량
   - 제품 식별은 제품 번호로 함
7. 회원의 물품 구매 내역을 관리하기 위해 Purchase History는 다음과 같은 정보를 요구함
   - 상품명, 금액
   - 가격은 회원이 구매한 해당 물품에 대한 누적 결제 금액임
   - 식별은 외부에서 외래키를 가져와 복합키를 만듦
     - Weak Entity
8. 개인 보관함 관리를 위해 Personal Locker entity는 다음과 같은 정보들을 요구함
   - 보관함 번호, 사용 중인 회원 번호, 이상 증세
   - 보관함 식별은 보관함 번호로 함
9. 추천인 시스템
   - 회원이 등록할 때 다른 회원을 0명부터 최대 1명까지 추천 가능
10. 회원이 물건을 구매하면 회원이 구매한 제품에 대한 누적 구매 이력이 업데이트 되야 함.
    - 회원이 구매하려는 제품의 재고가 부족하면 판매불가
11. 개인 보관함은 개인에게 할당되며, 하나의 개인 보관함이 여러명에게 할당될 수 없음. 단, 한 명의 회원은 여러 개의 개인 보관함을 사용할 수 있음
12. 트레이너는 프로그램을 개설하여, 멤버는 개설된 프로그램에 0개 혹은 여러 개를 등록할 수 있음
    - 하나의 프로그램 담당자는 트레이너 한 명임
    - 트레이너는 여러 개의 프로그램을 개설할 수 있음
    - 트레이너가 개설한 프로그램이 있을 때 트레이너가 일을 못하는 상황이면 그 트레이너가 개설했던 프로그램의 담당자는 우선적으로 트레이너

번호가 가장 빠른 트레이너가 됨

13. 트레이너는 담당 운동기구가 없을 수 있으며, 여러 운동 장비를 관리해야 할 수도 있음

- 운동기구의 담당 트레이너는 최소 1명 최대1명으로 고정되어 있음

III. ER Diagram



IV. DB Table 및 Schema

1. Member Table

MEMBER(*member_no, name, phone_number, gender, address, amount_of_payment,* ***recommender***)

- *Phase1, Phase 3 적용*

```
CREATE TABLE member (
    member_no INT NOT NULL,
    name VARCHAR(20) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    gender VARCHAR(1),
    address VARCHAR(150),
    amount_of_payment INT,
    recommender INT,
    PRIMARY KEY(member_no),
    FOREIGN KEY(recommender) REFERENCES member(member_no)
);
```

- Gender
  - F는 여성, M은 남성을 의미
- 추천인 시스템을 위한 자기 참조

■ 자기 참조로 인해 참조 제약 조건은 NO ACTION으로만 설정가능해서 참조 제약 조건 설정을 하지 않

```
INSERT INTO member VALUES(1, '이민준', '010-7728-1197', 'M', '서울특별시 관악구 신림로3가길 57(신림동, 서초그린아파트)', 300000, null);
INSERT INTO member VALUES(2, '김서연', '010-3281-3520', 'F', '서울특별시 관악구 대학5길 46(신림동, 래디앙아파트)', 200000, null);
INSERT INTO member VALUES(3, '박서준', '010-1577-1223', 'M', '서울특별시 관악구 광신길 160(신림동, 신림1단지주공아파트)', 250000, 2);
INSERT INTO member VALUES(4, '최서윤', '010-1292-1977', 'F', '서울특별시 관악구 호암로 417(신림동, 국제산장아파트)', 450000, 1);
INSERT INTO member VALUES(5, '김도윤', '010-1771-1901', 'M', '서울특별시 관악구 호암로 399(신림동, 삼성산주공아파트)', 1000000, 1);
INSERT INTO member VALUES(6, '이지우', '010-1231-3391', 'F', '서울특별시 관악구 난곡로 30(신림동, 관악산휴먼시아아파트)', 2000000, 1);
INSERT INTO member VALUES(7, '이하은', '010-0091-8172', 'F', '서울특별시 관악구 호암로 519(신림동, 벽산블루밍아파트)', 300000, 3);
INSERT INTO member VALUES(8, '박하준', '010-2881-4220', 'M', '서울특별시 관악구 호암로16길 5(신림동, 신림서초아파트)', 300000, 4);
INSERT INTO member VALUES(9, '강민서', '010-1121-3281', 'M', '서울특별시 관악구 호암로16길 11(신림동, 청화아파트)', 40000, 7);
INSERT INTO member VALUES(10, '우도현', '010-3812-3013', 'M', '서울특별시 관악구 신림로3가길 57(신림동, 서초그린아파트)', 40000, null);
```

2. Trainer Table

TRAINER(trainer_no, name, phone_number, gender, address, salary)

- Phase1

```
CREATE TABLE trainer(
    trainer_no INT NOT NULL,
    name VARCHAR(20) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    gender VARCHAR(1),
    address VARCHAR(150),
    salary INT,

    PRIMARY KEY(trainer_no)
);
```

```
INSERT INTO trainer VALUES(1, '이동현', '010-1283-1536', 'M', '서울특별시 관악구 신림로58길 62-5(신림동, 신안아파트)', 5000000);
INSERT INTO trainer VALUES(2, '김태윤', '010-3421-3456', 'M', '서울특별시 관악구 신림로48길 17-20(신림동, 신림건영4차아파트)', 4000000);
INSERT INTO trainer VALUES(3, '박이안', '010-1882-1223', 'M', '서울특별시 관악구 광신길 160(신림동, 신림1단지주공아파트)', 3300000);
INSERT INTO trainer VALUES(4, '최민규', '010-1292-1977', 'M', '서울특별시 서초구 서초대로1길 2(방배동, 방배한진로즈힐아파트)', 3000000);
INSERT INTO trainer VALUES(5, '김한결', '010-6678-3729', 'M', '서울특별시 서초구 서초대로33길 52(방배동, 방배 아크빌)', 3200000);
INSERT INTO trainer VALUES(6, '이재민', '010-8179-8347', 'M', '서울특별시 서초구 동광로32길 60(방배동, 필그린빌라)', 2500000);
INSERT INTO trainer VALUES(7, '이은찬', '010-1009-8562', 'M', '서울특별시 서초구 동광로 89-5(방배동, 방배3차이편한세상)', 3000000);
INSERT INTO trainer VALUES(8, '박윤호', '010-4491-4932', 'M', '서울특별시 관악구 호암로16길 5(신림동, 신림서초아파트)', 2500000);
INSERT INTO trainer VALUES(9, '강근', '010-9528-1313', 'M', '서울특별시 관악구 호암로16길 11(신림동, 청화아파트)', 2000000);
INSERT INTO trainer VALUES(10, '우아윤', '010-1133-9081', 'F', '서울특별시 강남구 개포로109길 5(개포동, 대치아파트)', 3300000);
```

3. Program Table

PROGRAM(program_no, program_name, program_type, price, current_number_of_member, limit_number_of_member)

- Phase 1 적용

```
CREATE TABLE program(
    program_no INT NOT NULL DEFAULT 0,
    program_name VARCHAR(100) NOT NULL,
    program_type VARCHAR(1),
    price INT,
    current_number_of_member INT,
    limit_number_of_member INT,

    PRIMARY KEY(program_no)
);
```

```
INSERT INTO program VALUES(1, 'Begginer Back', 'G', 250000, 1, 3);
INSERT INTO program VALUES(2, 'Advanced Chest', 'G', 400000, 1, 3);
INSERT INTO program VALUES(3, 'Beginner Shoulder', 'G', 250000, 1, 3);
INSERT INTO program VALUES(4, 'Beginner Triceps', 'G', 200000, 1, 3);
INSERT INTO program VALUES(5, 'ABS - basic', 'G', 100000, 1, 3);
INSERT INTO program VALUES(6, 'Beginner Biceps', 'G', 100000, 1, 5);
INSERT INTO program VALUES(7, 'Classic Physique - basic', 'P', 1000000, 1, 1);
INSERT INTO program VALUES(8, 'Classic Physique - advanced', 'P', 2000000, 1, 1);
INSERT INTO program VALUES(9, 'Diet Program', 'G', 300000, 1, 5);
INSERT INTO program VALUES(10, 'Strength - basic', 'G', 300000, 1, 2);
```

- Program_type: 'G'는 그룹, 'P'는 개인을 의미

4. Fitness Equipment Table

FITNESS_EQUIPMENT(serial_no, **trainer_no**, brand, equipment_name, abnormality_content, abnormality_occur_date)

- Phase1, phase 4 적용

```
CREATE TABLE fitness_equipment(
    serial_no INT NOT NULL,
    trainer_no INT NOT NULL default 1,
    brand varchar(20),
    equipment_name varchar(50),
    abnormality_content varchar(100),
    abnormality_occur_date date,

    PRIMARY KEY(serial_no),
    FOREIGN KEY(trainer_no) REFERENCES trainer(trainer_no)
    ON DELETE SET DEFAULT ON UPDATE CASCADE
);
```

- Trainer와 Fitness Equipment 관계에서 Fitness Equipment는 전체 참여이므로 Fitness Equipment에서 Trainer를 참조

```
INSERT INTO fitness_equipment VALUES(1211001, 1, 'Hammer Strength', 'Half rack', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1311891, 2, 'Hammer Strength', 'Pec Fly', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1102112, 3, 'Hammer Strength', 'Half rack', NULL, NULL);
INSERT INTO fitness_equipment VALUES(3314342, 4, 'Signature', 'Chest Press', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1134287, 5, 'Signature', 'Leg Curl', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1266113, 6, 'Signature', 'Leg Extension', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1177161, 7, 'Signature', 'Shoulder Press', NULL, NULL);
INSERT INTO fitness_equipment VALUES(3371841, 8, 'Pro2', 'Fixed Pulldown', '무게 추 고장', '2023-06-04');
INSERT INTO fitness_equipment VALUES(1192121, 9, 'Pro2', 'Tricep Arm Extension', NULL, NULL);
INSERT INTO fitness_equipment VALUES(1499181, 10, 'Pro2', 'Bicep Curl', '지지대 볼량', '2023-06-03');
```

5. Product Table

PRODUCT(product_no, price, product_name, quantity)

- Phase1 적용

```
CREATE TABLE product(
    product_no INT NOT NULL,
    price INT,
    product_name VARCHAR(50),
    quantity INT,

    PRIMARY KEY(product_no)
);
```

```
INSERT INTO product VALUES(1, 1800, '셀렉스 웨이프로틴', 12);
INSERT INTO product VALUES(2, 1900, '몬스터 에너지 울트라', 20);
INSERT INTO product VALUES(3, 1900, '삼대 오백 부스터', 12);
INSERT INTO product VALUES(4, 1000, '아이시스 물 500ML', 20);
INSERT INTO product VALUES(5, 1700, '파워에이드 마운틴 블라스트 500ML', 12);
INSERT INTO product VALUES(6, 1800, '게토레이 레몬 500ML', 12);
INSERT INTO product VALUES(7, 20000, 'WSF Lifting Strap', 5);
INSERT INTO product VALUES(8, 27000, 'Shield Lifiting Strap', 10);
INSERT INTO product VALUES(9, 12000, 'Blender Bottle', 10);
INSERT INTO product VALUES(10, 40000, 'Shield Lifting Belt', 7);
```

6. Purchase History Table

PURCHASE_HISTORY(**member_no**, **product_no**, prdocut_name, price)

- Phase 2, Phase 6 적용

```
CREATE TABLE purchase_history (
    member_no INT NOT NULL DEFAULT 0,
    product_no INT NOT NULL,
    product_name VARCHAR(50) NOT NULL,
    price INT, --회원이 구매한 구매물품 누적 금액

    PRIMARY KEY(member_no, product_no, product_name),
    FOREIGN KEY(member_no) REFERENCES member(member_no)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(product_no) REFERENCES product(product_no)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

- Weak Entity
- 프로젝트 제한 사항을 만족하기 위해 relation을 위한 새로운 매개체 테이블을 만들지 않고 Purchase History에서 연관관계를 유지함

```
INSERT INTO purchase_history VALUES(1, 1, '셀렉스 웨이 프로틴', 1800);
INSERT INTO purchase_history VALUES(1, 2, '몬스터 에너지 울트라', 1900);
INSERT INTO purchase_history VALUES(1, 8, 'Shield Lifiting Strap', 27000);
INSERT INTO purchase_history VALUES(2, 9, 'Blender Bottle', 12000);
INSERT INTO purchase_history VALUES(2, 1, '셀렉스 웨이 프로틴', 1800);
INSERT INTO purchase_history VALUES(4, 10, 'Shield Lifting Belt', 40000);
INSERT INTO purchase_history VALUES(4, 1, '셀렉스 웨이 프로틴', 1800);
INSERT INTO purchase_history VALUES(6, 3, '삼대 오백 부스터', 1900);
INSERT INTO purchase_history VALUES(7, 1, '셀렉스 웨이 프로틴', 1800);
INSERT INTO purchase_history VALUES(8, 1, '셀렉스 웨이 프로틴', 1800);
```

7. Personal Locker Table

PERSONAL_LOCKER(locker_no, abnormality_content, **member_no**)

- Phase 1, Phase 4 적용

```
CREATE TABLE personal_locker (
    locker_no INT NOT NULL,
    abnormality_content VARCHAR(100),
    member_no INT,
    PRIMARY KEY(locker_no),
    FOREIGN KEY(member_no) REFERENCES member(member_no)
    ON DELETE SET NULL ON UPDATE CASCADE
);
```

- Member와 Personal Locker는 N : 1관계 이므로 Personal Locker에서 Member를 참조하도록 함

```
INSERT INTO personal_locker VALUES(1, null, null);
INSERT INTO personal_locker VALUES(2, null, 2);
INSERT INTO personal_locker VALUES(3, null, 1);
INSERT INTO personal_locker VALUES(4, null, 4);
INSERT INTO personal_locker VALUES(5, null, 3);
INSERT INTO personal_locker VALUES(6, null, null);
INSERT INTO personal_locker VALUES(7, null, 5);
INSERT INTO personal_locker VALUES(8, null, null);
INSERT INTO personal_locker VALUES(9, null, null);
INSERT INTO personal_locker VALUES(10, null, null);
```

8. Participate Table

PARTICIPATE(member_no, trainer_no, program_no)

- Relation에서 Table로 변환된 형태
- Phase 6를 적용

```
CREATE TABLE participate(
    member_no INT NOT NULL,
    trainer_no  INT NOT NULL DEFAULT 1,
    program_no INT NOT NULL DEFAULT 0,

    FOREIGN KEY (member_no) REFERENCES member(member_no)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (trainer_no) REFERENCES trainer(trainer_no)
    ON DELETE SET DEFAULT ON UPDATE CASCADE,
    FOREIGN KEY (program_no) REFERENCES program(program_no)
    ON DELETE SET DEFAULT ON UPDATE CASCADE,

    PRIMARY KEY(member_no, trainer_no, program_no)
);

INSERT INTO participate VALUES(1, 8, 1);
INSERT INTO participate VALUES(1, 6, 2);
INSERT INTO participate VALUES(2, 6, 3);
INSERT INTO participate VALUES(3, 4, 4);
INSERT INTO participate VALUES(4, 4, 5);
INSERT INTO participate VALUES(4, 7, 6);
INSERT INTO participate VALUES(5, 1, 7);
INSERT INTO participate VALUES(6, 1, 8);
INSERT INTO participate VALUES(7, 2, 9);
INSERT INTO participate VALUES(8, 10, 10);
```

9. 전체 Schema

MEMBER(member_no, name, phone_number, gender, address, amount_of_payment, **recommender**)

TRAINER(trainer_no, name, phone_number, gender, address, salary)

PROGRAM(program_no, program_name, program_type, price, current_number_of_member, limit_number_of_member)

FITNESS_EQUIPMENT(serial_no, **trainer_no**, brand, equipment_name, abnormality_content, abnormality_occur_date)

PRODUCT(product_no, price, product_name, quantity)

PERSONAL_LOCKER(locker_no, abnormality_content, **member_no**)

PARTICIPATE(**member_no**, **trainer_no**, **program_no**)

PURCHASE_HISTORY(**member_no**, **product_no**, prdocut_name, price)

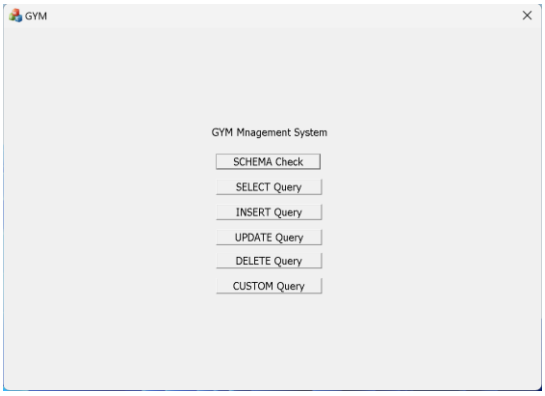
V. 프로그램의 구현 단계

기능	구현단계
Schema Check	DB에 생성된 8개 Table에 대한 Schema를 화면에 표시함 완벽히 작동
SELECT Query	5개중 5개 구현완료, 에러 처리 완료
INSERT Query	5개중 5개 구현완료, 세세한 예외처리까진 아니더라도 프로그램이 멈추는 일이 없도록 예외처리
DELETE Query	4개중 4개 구현완료, 에러 처리 완료
UPDATE Query	4개중 4개 구현완료, 에러 처리 완료
Custom Query	6개중 6개 구현 완료, 에러 처리 완료
JOIN	2개 테이블 JOIN 4개, 3개 테이블 JOIN 2개

VI. 프로그램 실행

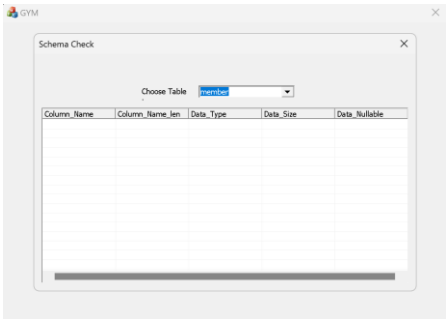
- MFC 기반으로 프로그램 작성

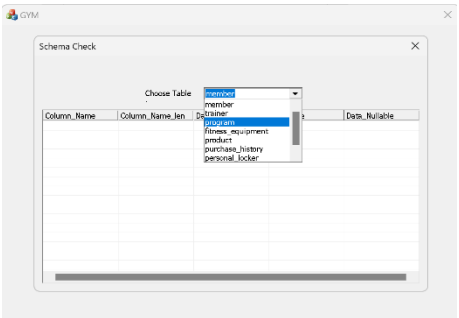
1. 초기 화면

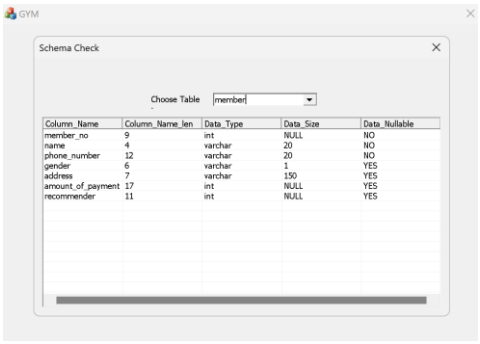


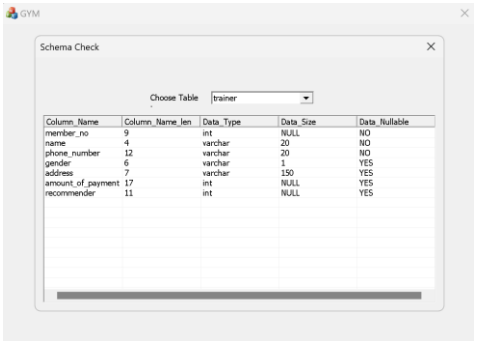
- 6개의 메뉴버튼
- 각 메뉴버튼 클릭 시 다이얼로그 창 생성

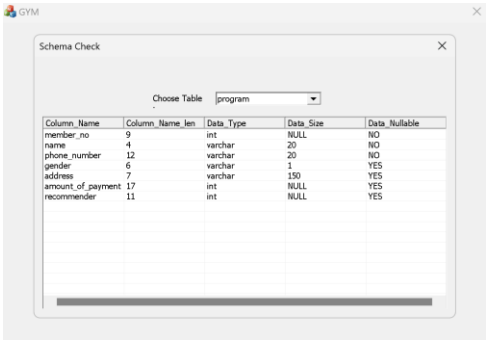
2. SCHEMA Check

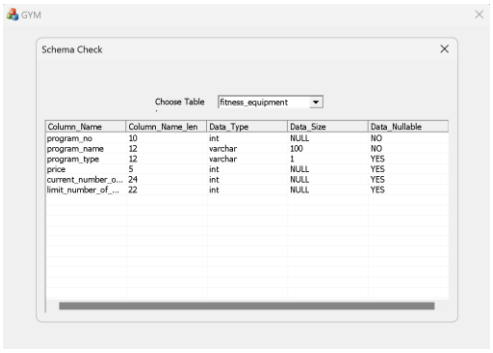


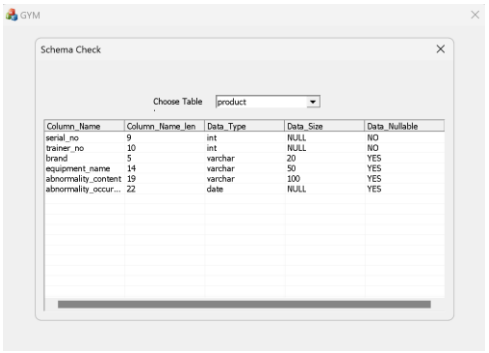


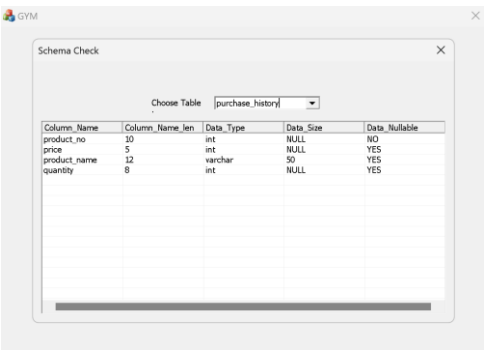


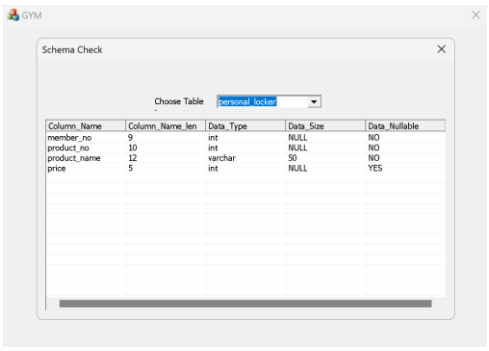


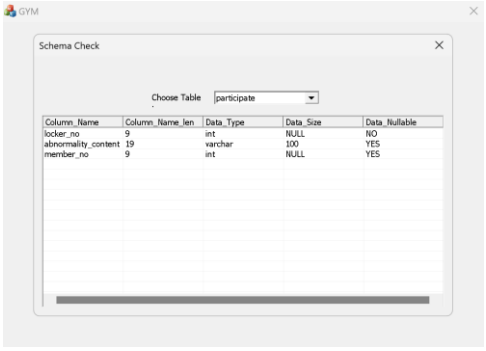












- 콤보 박스에서 원하는 Table의 Schema를 선택가능
- 콤보 박스에서 Schema를 선택시 List Control에 스키마 정보 표

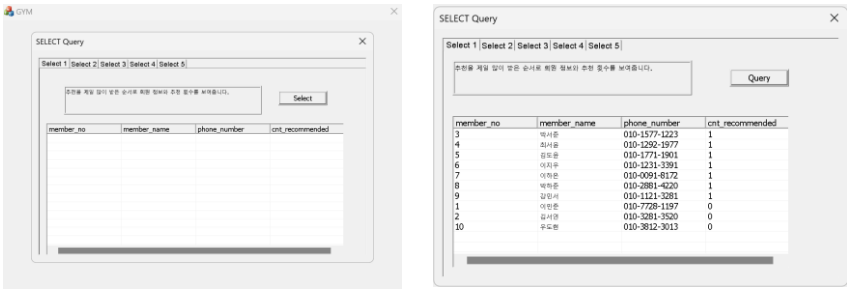
현

```
SELECT Column_Name,
    len(COLUMN_NAME) AS Column_Name_len,
    Data_Type,
    CHARACTER_MAXIMUM_LENGTH AS Data_Size,
    IS_NULLABLE As Data_Nullable
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'fitness_equipment';
```

- TABLE 별로 TABLE_NAME만 변함

3. SELECT Query

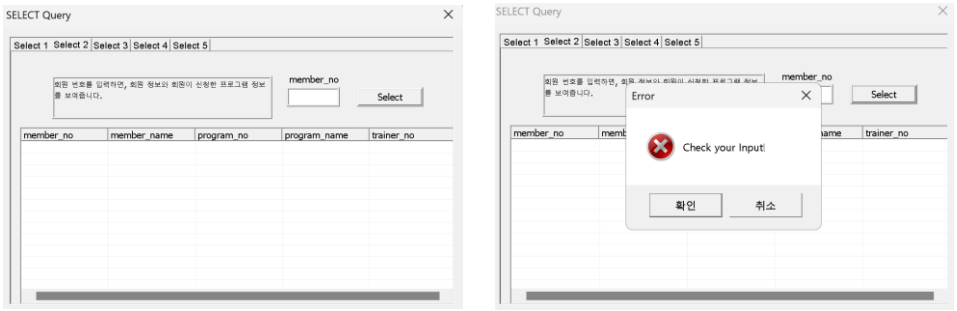
- Tab Controller로 시나리오 선택 가능
- Senario1: 추천을 제일 많이 받은 순서로 회원 정보와 추천 횟수를 보여줍니다.



  - 버튼 클릭시 해당 시나리오 결과가 List에 보여짐

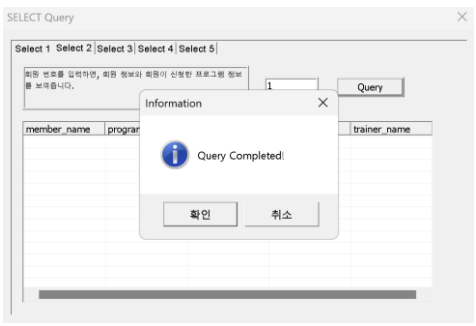
```
SELECT member_no, name, phone_number, COUNT(recommender) AS RecommendationCount
FROM member
GROUP BY member_no, Name, phone_number
ORDER BY RecommendationCount DESC, member_no ASC;
```

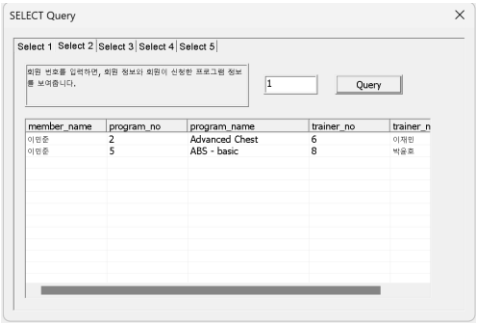
- Senario2: 회원 번호를 입력하면 회원 정보와 회원이 신청한 프로그램 정보를 보여줌



  - 회원 번호를 입력하지 않고 Select 버튼을 누르면 에러처리

- 회원 번호에 숫자가 아닌 것들을 넣은 상태로 Select 버튼을 누르면 에러처리
- 회원 번호에 존재하지 않는 회원 번호를 넣은 상태에서 Select 버튼을 누르면 에러처리

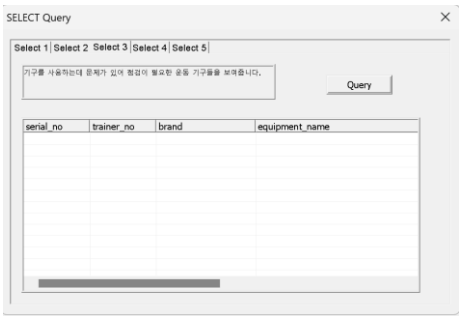


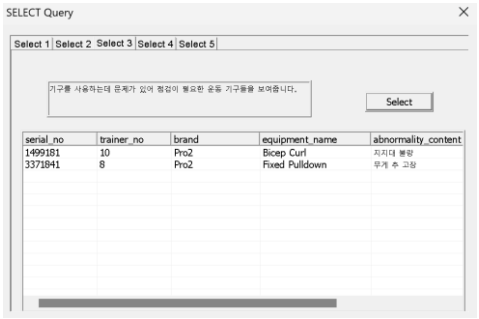


- 입력창에 회원 번호 1를 입력하고 Query 버튼을 눌러 실행
- 3개의 Table을 JOIN하는 Senario
  - member table, program table, participate table

```
SELECT m.member_no, m.name, p.program_no, p.program_name, pt.trainer_no
FROM member AS m
JOIN participate AS pt ON m.member_no = pt.member_no
JOIN program AS p ON pt.program_no = p.program_no
WHERE m.member_no = 1;
```

- Senario3: 문제가 있는 운동 기구들을 보여주는 시나리오

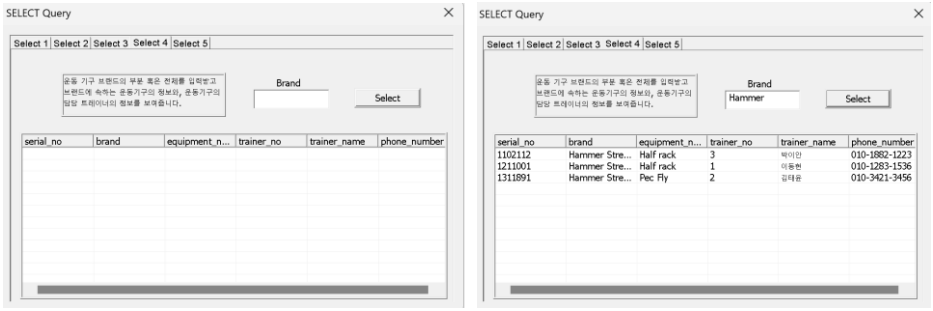




- Select 버튼을 눌러 결과를 보여줌

```
SELECT * FROM fitness_equipment
WHERE abnormality_content is NOT NULL ;
```

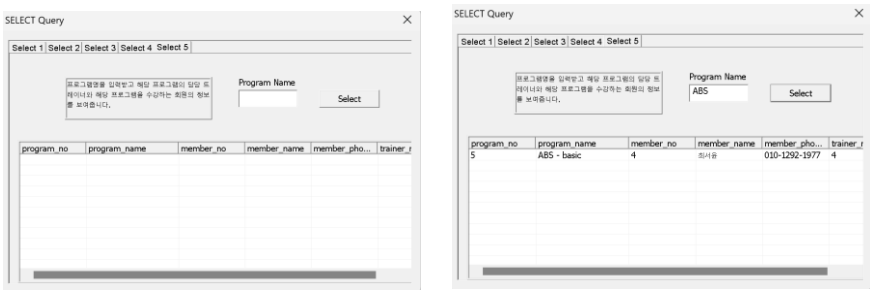
- Senario4: 운동 기구의 브랜드를 입력 받고 해당 브랜드에 속하는 운동 기구와 트레이너 정보를 보여줌



- 입력창에 "Hammer"를 입력하고 버튼을 눌러 결과를 보여줌
- 2개의 Table을 JOIN하는 시나리오

```
SELECT fe.serial_no, fe.brand, fe.equipment_name, t.trainer_no, t.name, t.phone_number
FROM fitness_equipment AS fe
JOIN trainer AS t ON fe.trainer_no = t.trainer_no
WHERE fe.brand LIKE '%HAMMER%'
ORDER BY serial_no;
```

- Senario5: 프로그램명을 입력 받고 해당 프로그램의 담당 트레이너와 해당 프로그램을 수강하는 회원의 정보를 보여줌

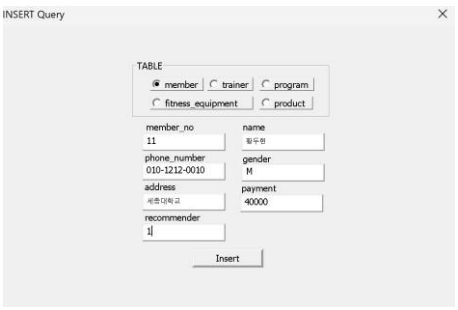


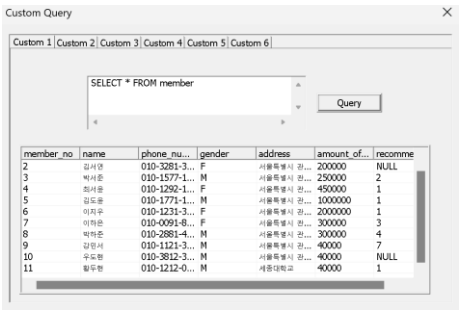
- 입력창에 "ABS"를 입력하고 버튼을 눌러 결과를 보여줌
- 3개의 Table을 JOIN하는 Scenario

```
SELECT DISTINCT p.program_no, p.program_name, m.member_no, m.name, m.phone_number, pt.trainer_no
FROM program AS p
JOIN participate As pt ON p.program_no = pt.program_no
JOIN member m ON pt.member_no = m.member_no
WHERE p.program_name LIKE '%ABS%'
ORDER BY program_no;
```

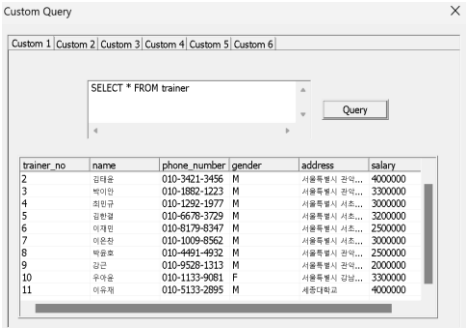
4. INSERT Query

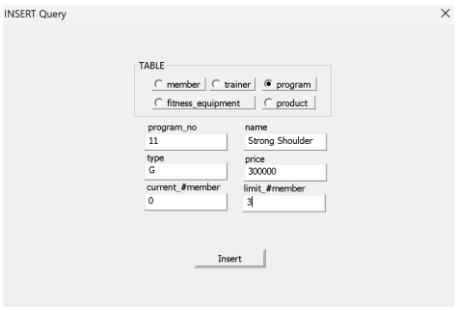

INSERT Query
TABLE
member
trainer
program
fitness_equipment
product

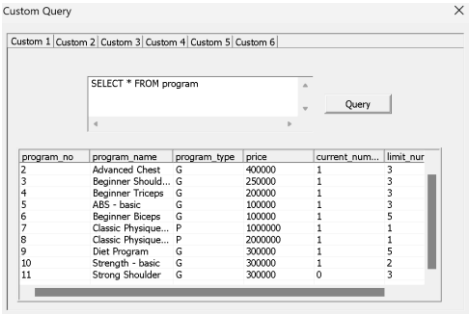

INSERT Query
TABLE
member
trainer
program
fitness_equipment
product
member_no
11
name
phone_number
010-1212-0010
gender
M
address
payment
40000
recommender
1
Insert

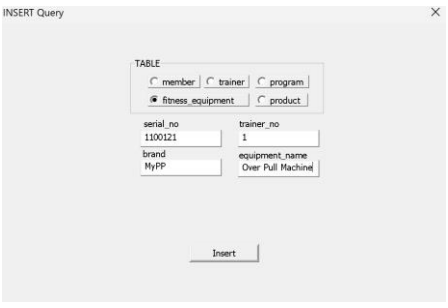

Custom Query
Custom 1
Custom 2
Custom 3
Custom 4
Custom 5
Custom 6
SELECT * FROM member
Query
member_no
name
phone_nu...
gender
address
amount_of...
recomme

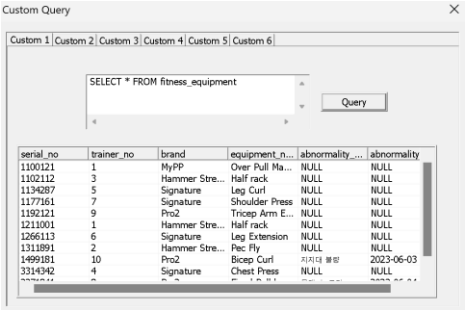

GYM
INSERT Query
TABLE
member
trainer
program
fitness_equipment
product
trainer_no
11
name
phone_number
010-5133-2895
gender
M
address
salary
4000000
Insert

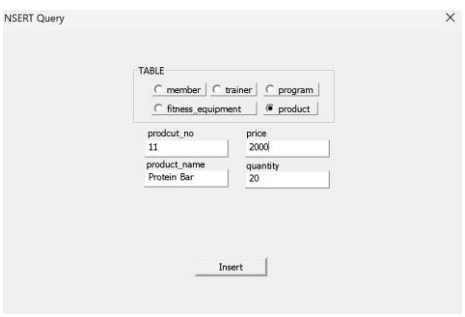

Custom Query
Custom 1
Custom 2
Custom 3
Custom 4
Custom 5
Custom 6
SELECT * FROM trainer
Query
trainer_no
name
phone_number
gender
address
salary

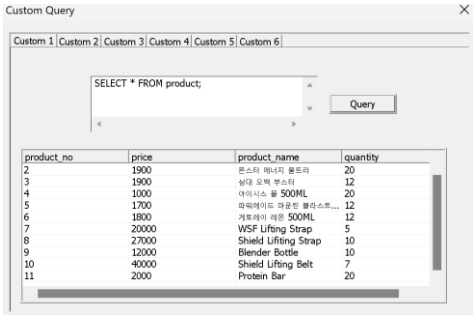

INSERT Query
TABLE
member
trainer
program
fitness_equipment
product
program_no
11
name
Strong Shoulder
type
G
price
300000
current_#member
0
limit_#member
3
Insert


Custom Query
Custom 1
Custom 2
Custom 3
Custom 4
Custom 5
Custom 6
SELECT * FROM program
Query
program_no
program_name
program_type
price
current_num...
limit_nur


INSERT Query
TABLE
member
trainer
program
fitness_equipment
product
serial_no
1100121
trainer_no
1
brand
MyPP
equipment_name
Over Pull Machine
Insert


Custom Query
Custom 1
Custom 2
Custom 3
Custom 4
Custom 5
Custom 6
SELECT * FROM fitness_equipment
Query
serial_no
trainer_no
brand
equipment_n...
abnormality_...
abnormality



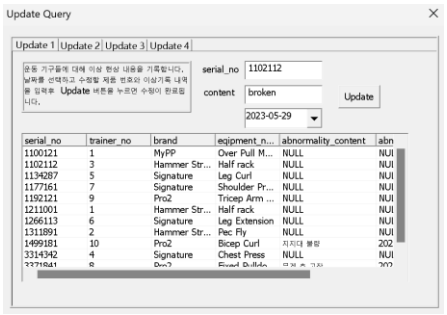


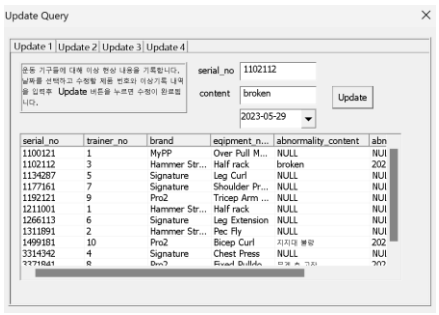
- Radio button으로 삽입을 실행할 테이블을 선택
- 입력창에 데이터를 입력하고 Insert 버튼 클릭
- 입력창에 데이터를 비워 두면 쿼리 요청시 NULL값으로 들어감
- 삽입후 Custom query1에서 테이블 전체를 조회

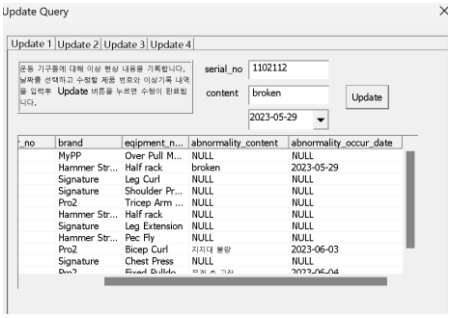
```
INSERT INTO member VALUES(11, '황두현', '010-1212-0010', 'M', '세종대학교', 40000, 1);
INSERT INTO trainer VALUES(11, '이유재', '010-5133-2895', 'M', '세종대학교', 4000000);
INSERT INTO program VALUES(11, 'Strong Shoulder', 'G', 300000, 0, 3);
INSERT INTO fitness_equipment VALUES(1100121, 1, 'MyPP', 'Over Pull Machine', NULL, NULL);
INSERT INTO product VALUES(11, 2000, 'Protein Bar', 20);
```

5. UPATE Query

- Senario1: 운동 기구에 대한 시리얼 번호와, 이상 현상 내용, 날짜를 입력하여 업데이트



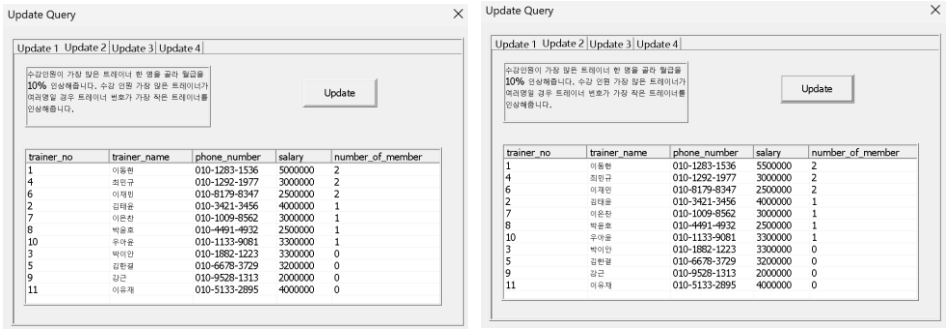




- 데이터 입력 후 업데이트 버튼 클릭

```sql
UPDATE GYM.dbo.fitness_equipment
SET abnormality_content = 'broken', abnormality_occur_date = '2023-05-29' --입력받은 문자열들이 들어감
WHERE serial_no = 1102112;
```

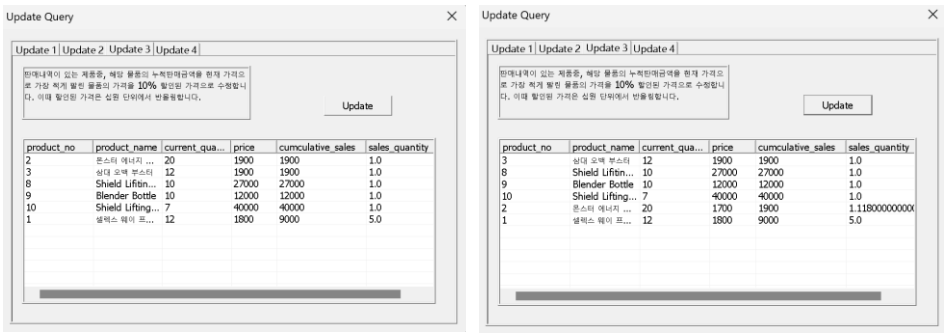
- Senario2: 수강인원이 제일 많은 트레이너의 급여를 10%인상해주는 시나리오



  - 기본적으로 트레이너 정보를 수강 인원수 내림차순으로 보여줌
  - 2개의 Table을 JOIN하는 Senario
  - Update 버튼을 눌러 결과를 바로 확인가능

```sql
UPDATE t
SET t.salary = t.salary * 1.1
FROM trainer AS t
JOIN (
    SELECT TOP 1 pt.trainer_no, COUNT(*) AS number_of_member
    FROM participate AS pt
    GROUP BY pt.trainer_no
    ORDER BY number_of_member DESC, trainer_no
) AS sub ON t.trainer_no = sub.trainer_no;
```

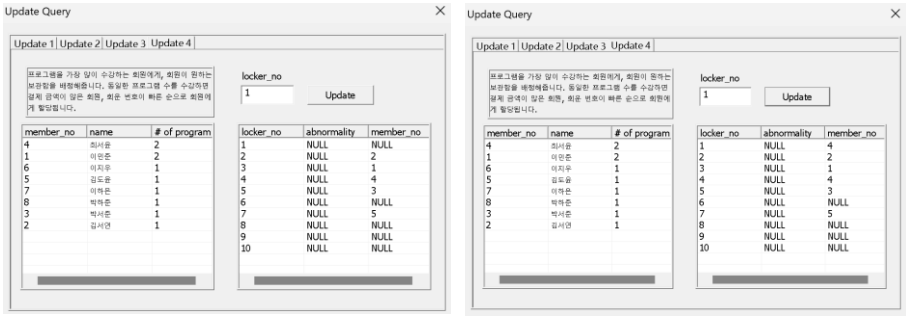
- Senario3: 판매내역이 있는 제품중, 해당 물품의 누적판매금액을 현재 가격으로 가장 적게 팔린 물품의 가격을 10% 할인된 가격으로 수정하는 시나리오



- Update 버튼을 눌러 실행
- 1개 팔린 제품들이 여러 개 있지만, 몬스터 에너지 음료의 제품 번호가 가장 적어 업데이트됨
- 2개의 Table을 JOIN하는 Scenario

```
UPDATE product
SET price = ROUND(price * 0.9, -2)
WHERE product_no = (SELECT TOP 1 ph.product_no FROM purchase_history AS ph
                    JOIN product AS p ON p.product_no = ph.product_no
                    GROUP BY ph.product_no, ph.product_name, p.price, p.quantity
                    ORDER BY(convert(float, sum(ph.price)) / p.price), product_no);
```

- Senario4: 프로그램을 가장 많이 수강하는 회원에게, 회원이 원하는 보관함을 배정해줌. 동일한 프로그램 수를 수강하면 결제 금액이 많은 회원, 회운 번호이 빠른 순으로 회원에게 할당됨.

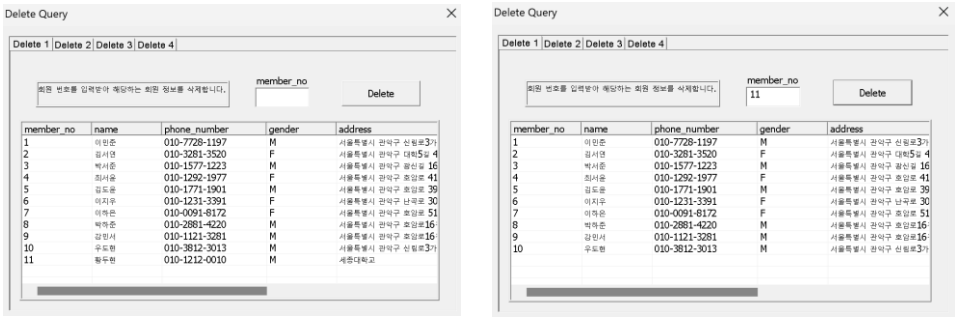


- 4번 회원이 조건에 부합하여 1번 라커룸이 4번 회원에게 할당됨
- 2개의 Table을 JOIN하는 Scenario

```
UPDATE personal_locker
SET member_no = (   SELECT TOP 1 m.member_no
                    FROM member AS m
                    JOIN participate As p ON m.member_no = p.member_no
                    GROUP BY m.member_no, m.name, m.amount_of_payment
                    ORDER BY COUNT(*) DESC, m.amount_of_payment DESC, member_no)
WHERE locker_no = 1; -- 사용자 입력으로 locker_no를 받음
```

6. DELETE Query

- Senario1: 회원 번호를 입력받아 해당 회원을 DB에서 삭제

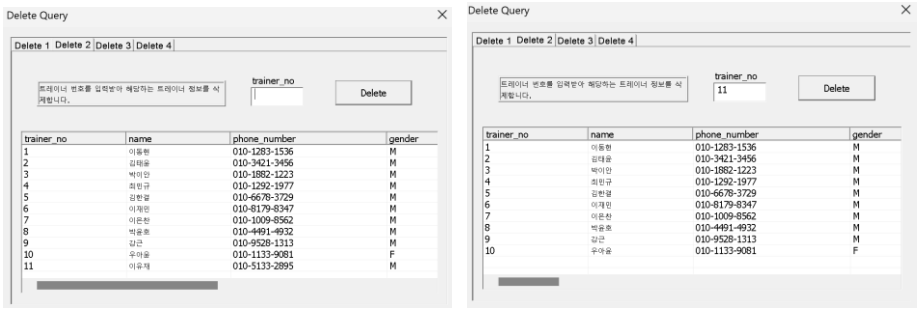


- 입력창에 11입력후 삭제버튼 클릭
- 자기 참조로 인해 삭제할 회원을 참조하고 있는 경우 NULL로 바꾸는 업데이트 작업을 하고 삭제함

```
UPDATE member
SET recommender = NULL
WHERE recommender = 11; --사용자 입력

DELETE FROM member WHERE member_no = 11; --사용자 입력
```

- Senario2: 트레이너 번호를 입력받아 해당 트레이너를 DB에서 삭제



- 입력창에 11입력후 삭제 버튼 클릭
- 11번 트레이너가 삭제됨
- 트레이너를 삭제하기전 트레이너를 참조하고 있던 fitness_equipment 튜플, 과 pariticipate 튜플에 대해 트레이너를 바꿔줌

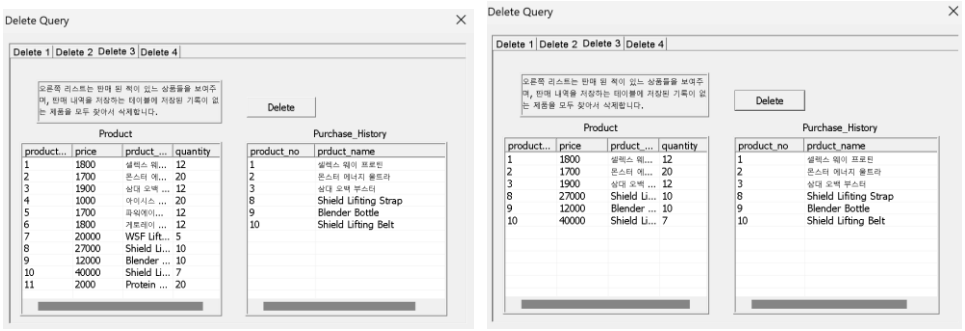
```
DECLARE @P_LOWEST_TRAINER_NO INT,
        @P_DELETE_TRAINER_NO INT

SET @P_DELETE_TRAINER_NO = 1 --사용자 입력
SET @P_LOWEST_TRAINER_NO = (SELECT TOP 1 trainer_NO FROM trainer
                            WHERE trainer_no <> @P_DELETE_TRAINER_NO
                            ORDER BY trainer_no);
UPDATE fitness_equipment
SET trainer_no = @P_LOWEST_TRAINER_NO
WHERE trainer_no = @P_DELETE_TRAINER_NO;

UPDATE participate
SET trainer_no = @P_LOWEST_TRAINER_NO
WHERE trainer_no = @P_DELETE_TRAINER_NO;

DELETE FROM trainer WHERE trainer_no = @P_DELETE_TRAINER_NO;
```

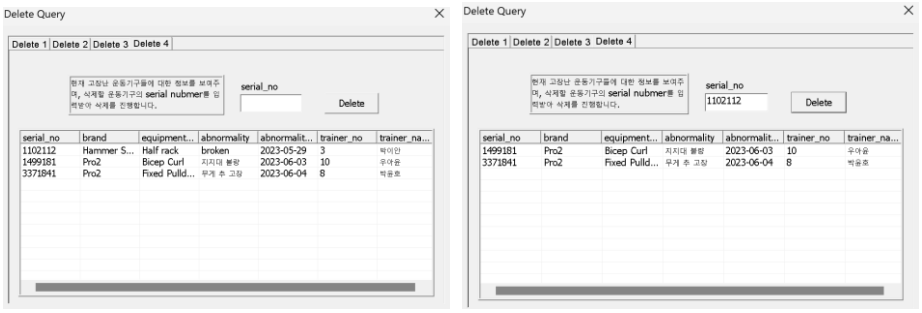
- Senario3: 판매 기록이 없는 판매 물품들을 삭제하는 시나리오



  - 한 화면에서 왼쪽리스트는 판매가능한 제품들의 리스트이고 오른쪽 리스트는 판매내역이 있는 리스트임
  - Delete 버튼을 누르면 쿼리가 진행됨

```
DELETE FROM product
WHERE product_no NOT IN (
    SELECT DISTINCT product_no
    FROM purchase_history
)
```

- Senario4: 현재 고장 난 운동기구들에 대한 정보를 보여주고 운동기구의 serial number를 입력 받아 삭제하는 시나리오

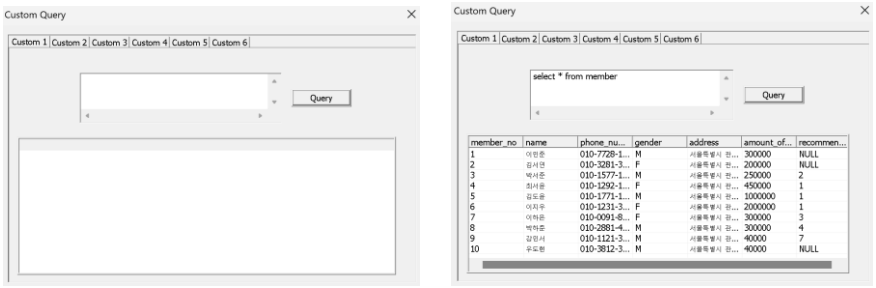


- 1102112를 입력하여 삭제를 진행함

```
DELETE fitness_equipment
WHERE abnormality_content is NOT NULL AND serial_no = 1102112 -- 사용자 입력
```

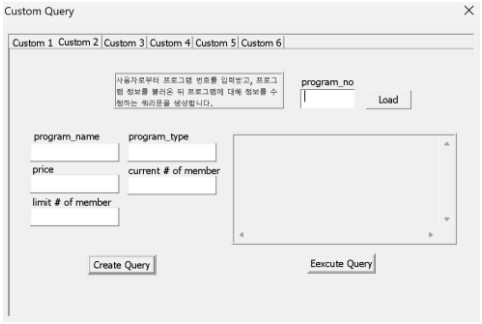
7. CUSTOM Query

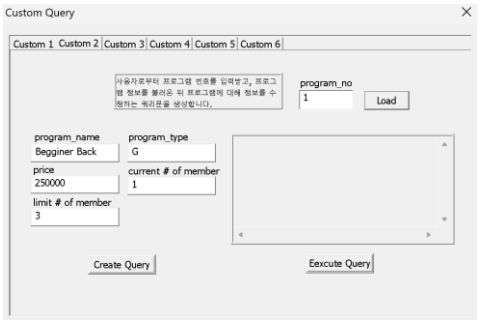
- Custom 1: 사용자가 원하는 query문을 입력할 수 있음



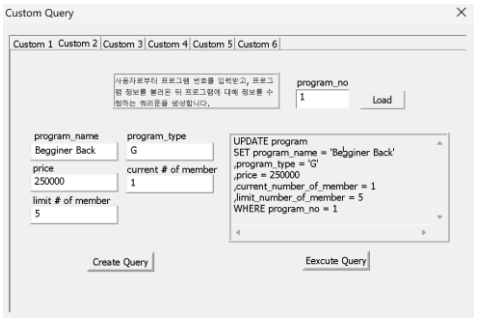
  - Edit Control에 값을 입력한 뒤 Query 버튼을 눌러 사용자 입력 쿼리 실행
  - SELECT 문을 요청했을 시 LIST CONTROL이 쿼리 결과에 맞춰 업데이트 됨
  - UPDATE, DELETE도 query문도 가능

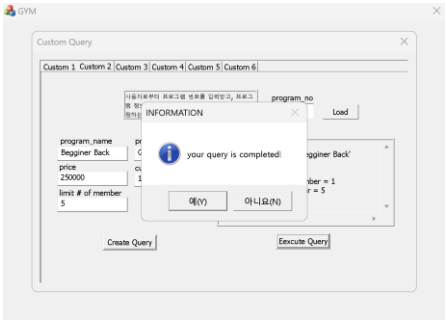
- Custom 2: 사용자로부터 프로그램 번호를 입력 받아 해당 프로그램의 기존 정보를 불러온 뒤 수정하고 Query를 만들어 실행할 수 있음



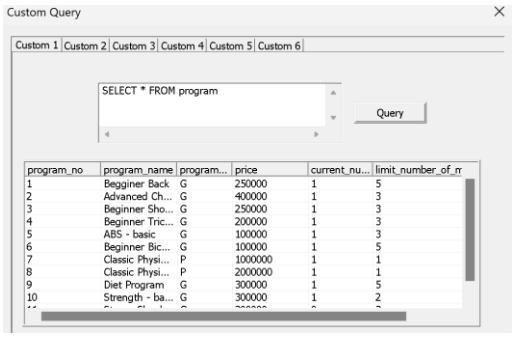


- 1을 입력하고 Load 버튼을 눌러 해당 프로그램 데이터를 불러옴

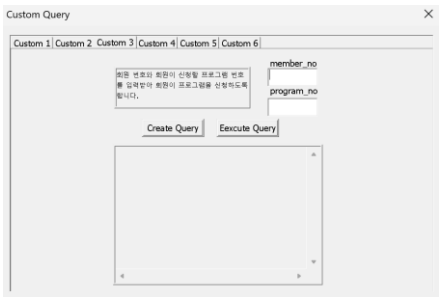


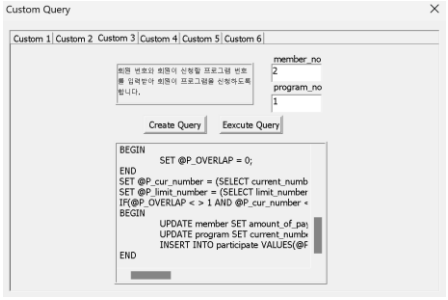


- Limit # of member를 1에서 5로 수정하고 CREATE Query를 통해 Query를 만들고 보여줌
- EXECURE Query버튼을 통해 생성한 쿼리를 실행함



- Custom 3: 회원 번호와 프로그램 번호를 입력 받아 회원이 프로그램에 등록하도록 함





■ member_no에 2를 넣고 program_no에 1을 넣어 쿼리문을 만들어 보여줌

```
DECLARE @P_TRAINER_NO INT,
        @P_MEMBER_NO INT,
        @P_PROGRAM_NO INT,
        @P_OVERLAP BIT,
        @P_cur_number INT,
        @P_limit_number INT

SET @P_PROGRAM_NO = 1; --사용자 입력
SET @P_MEMBER_NO = 2; --사용자 입력
SET @P_TRAINER_NO = (SELECT DISTINCT trainer_no FROM participate WHERE program_no = @P_PROGRAM_NO);

IF EXISTS(
    SELECT 1
    FROM participate
    WHERE member_no = @P_MEMBER_NO AND program_No = @P_PROGRAM_NO
)
BEGIN
    SET @P_OVERLAP = 1;
END
ELSE
BEGIN
    SET @P_OVERLAP = 0;
END

SET @P_cur_number = (SELECT current_number_of_member FROM program WHERE program_no = @P_PROGRAM_NO);
SET @P_limit_number = (SELECT limit_number_of_member FROM program WHERE program_no = @P_PROGRAM_NO);
IF (@P_OVERLAP <> 1 AND @P_cur_number < @P_limit_number)
BEGIN
    UPDATE member SET amount_of_payment = amount_of_payment + (SELECT price FROM program WHERE program_no = @P_PROGRAM_NO) WHERE member_no = @P_MEMBER_NO;
    UPDATE program SET current_number_of_member = current_number_of_member + 1 WHERE program_no = @P_PROGRAM_NO;
    INSERT INTO participate VALUES(@P_MEMBER_NO, @P_TRAINER_NO, @P_PROGRAM_NO);
END
```

■ 생성된 쿼리문

– 쿼리 실행 전

member_no	name	phone_nu...	gender	address	amount_of...	recommen...
1	이민준	010-7728-1...	M	서울특별시 관...	300000	NULL
2	김서연	010-3281-3...	F	서울특별시 관...	200000	NULL

program_no	program_name	program_type	price	current_num...	limit_numbe
1	Begginer Back	G	250000	1	5

member_no	trainer_no	program_no
1	6	2
1	8	1
2	6	3
3	4	4
4	4	5
4	7	6
5	1	7
6	1	8
7	2	9
8	10	10

■ 2번 회원의 결제 금액은 20만원이고

■ 1번 프로그램의 현재 수강 인원은 1명

■ participate 테이블에 회원 2와 프로그램 1사이의 관계는 없음

– 쿼리 실행 후

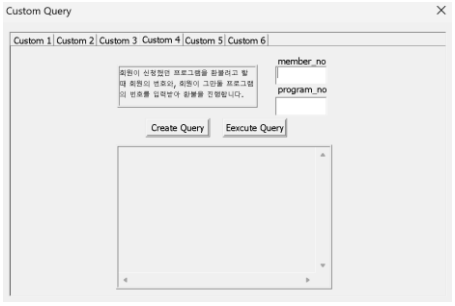
member_no	name	phone_nu...	gender	address	amount_of...	recommen...
1	이민준	010-7728-1...	M	서울특별시 관...	300000	NULL
2	김서연	010-3281-3...	F	서울특별시 관...	450000	NULL

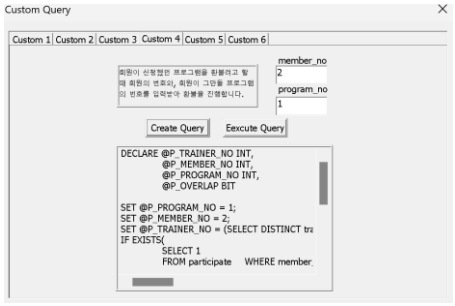
program_no	program_name	program_type	price	current_num...	limit_numbe
1	Begginer Back	G	250000	2	5

member_no	trainer_no	program_no
1	6	2
1	8	1
2	6	3
2	8	1

- 2번 회원의 결제 금액은 프로그램 1번의 수강료인 25만원 증가
- 1번 프로그램의 수강 인원이 1 증가하여 2가 됨
- 2번 회원과 1번 프로그램 간의 관계가 participate 테이블에 반영됨

- Custom 4: 회원 번호와 프로그램 번호를 입력 받아 회원이 프로그램을 환불하도록 함





- member_no에 2를 넣고 program_no에 1을 넣어 쿼리문을 만들어 보여줌

```
DECLARE @P_TRAINER_NO INT,
        @P_MEMBER_NO INT,
        @P_PROGRAM_NO INT,
        @P_OVERLAP BIT

SET @P_PROGRAM_NO = 1; --사용자 입력
SET @P_MEMBER_NO = 2; --사용자 입력
SET @P_TRAINER_NO = (SELECT DISTINCT trainer_no FROM participate WHERE program_no = @P_PROGRAM_NO);

IF EXISTS(
    SELECT 1
    FROM participate
    WHERE member_no = @P_MEMBER_NO AND program_No = @P_PROGRAM_NO
)
BEGIN
    SET @P_OVERLAP = 1;
END
ELSE
BEGIN
    SET @P_OVERLAP = 0;
END

IF (@P_OVERLAP = 1)
BEGIN
    UPDATE member SET amount_of_payment = amount_of_payment - (SELECT price FROM program WHERE program_no = @P_PROGRAM_NO) WHERE member_no = @P_MEMBER_NO;
    UPDATE program SET current_number_of_member = current_number_of_member - 1 WHERE program_no = @P_PROGRAM_NO;
    DELETE participate WHERE member_no = @P_MEMBER_NO AND trainer_no = @P_TRAINER_NO AND  program_no = @P_PROGRAM_NO;
END
```

- 생성된 쿼리문

– 쿼리 실행 전

member_no	name	phone_nu...	gender	address	amount_of...	recommen...
1	이민준	010-7728-1...	M	서울특별시 관...	300000	NULL
2	김서연	010-3281-3...	F	서울특별시 관...	450000	NULL

program_no	program_name	program_type	price	current_num...	limit_numbe
1	Begginer Back	G	250000	2	5

member_no	trainer_no	program_no
1	6	2
1	8	1
2	6	3
2	8	1

- 2번 회원의 결제 금액은 45만원
- 1번 프로그램의 현재 수강 인원은 2명
- 2번 회원과 1번 프로그램 간의 관계가 participate 테이블에 있음

– 쿼리 실행 후

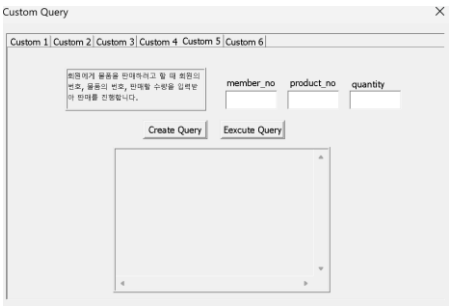
member_no	name	phone_nu...	gender	address	amount_of...	recommen...
1	이민준	010-7728-1...	M	서울특별시 관...	300000	NULL
2	김서연	010-3281-3...	F	서울특별시 관...	200000	NULL

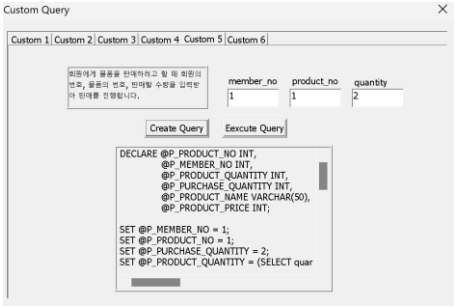
program_no	program_name	program_type	price	current_num...	limit_numbe
1	Begginer Back	G	250000	1	5

member_no	trainer_no	program_no
1	6	2
1	8	1
2	6	3
3	4	4
4	4	5
4	7	6
5	1	7
6	1	8
7	2	9
8	10	10

- 2번 회원의 결제 금액은 45만원에서 1번 프로그램의 수강료 25만원 전액환불 받아 20만원이 됨
- 1번 프로그램의 현재 수강 인원은 1명
- participate 테이블에 회원 2와 프로그램 1사이의 관계는 음

- Custom 5: 회원 번호, 상품 번호, 판매 수량을 입력 받아 회원에게 판매





- member_no에 1을 넣고 product_no에 1을 넣고 quantity에 2를 넣어 쿼리문을 만들어 보여줌

```
DECLARE @P_PRODUCT_NO INT,
        @P_MEMBER_NO INT,
        @P_PRODUCT_QUANTITY INT,
        @P_PURCHASE_QUANTITY INT,
        @P_PRODUCT_NAME VARCHAR(50),
        @P_PRODUCT_PRICE INT;

SET @P_MEMBER_NO = 1;  --사용자 입력
SET @P_PRODUCT_NO = 1; --사용자 입력
SET @P_PURCHASE_QUANTITY = 2; -- 사용자 입력

SET @P_PRODUCT_QUANTITY = (SELECT quantity
                    FROM product
                    WHERE product_no = @P_PRODUCT_NO)

IF(@P_PRODUCT_QUANTITY >= @P_PURCHASE_QUANTITY)
BEGIN
    SET @P_PRODUCT_NAME = (SELECT product_name
                           FROM product
                           WHERE product_no = @P_PRODUCT_NO);
    SET @P_PRODUCT_PRICE = (SELECT price
                           FROM product
                           WHERE product_no = @P_PRODUCT_NO);
    IF EXISTS(SELECT 1
              FROM purchase_history
              WHERE member_no = @P_MEMBER_NO AND product_no = @P_PRODUCT_NO)
    BEGIN
        UPDATE purchase_history
        SET price = ph.price + @P_PURCHASE_QUANTITY * p.price
        FROM purchase_history AS ph
        JOIN product AS p ON ph.product_no = p.product_no
        WHERE ph.product_no = @P_PRODUCT_NO;
    END
    ELSE
    BEGIN
        INSERT INTO purchase_history VALUES(@P_MEMBER_NO, @P_PRODUCT_NO, @P_PRODUCT_NAME, @P_PRODUCT_PRICE * @P_PURCHASE_QUANTITY)
    END

    UPDATE product
    SET quantity = @P_PRODUCT_QUANTITY - @P_PURCHASE_QUANTITY
    WHERE product_no = @P_PRODUCT_NO
END
```

- 생성된 쿼리문

– 쿼리 실행 전

product_no	price	product_name	quantity
1	1800	셀렉스 웨이프로틴	12

member_no	product_no	product_name	price
1	1	셀렉스 웨이 프로틴	1800

- product table에서 1번 상품의 수량은 12개
- purchase_history에서 1번 회원의 1번 제품 누적 구매 금액은 1800원

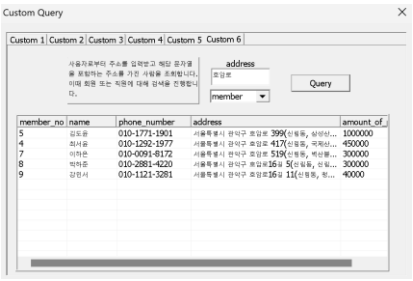
– 쿼리 실행 후

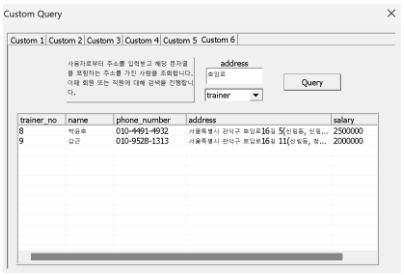
product_no	price	product_name	quantity
1	1800	셀렉스 웨이프로틴	10

member_no	product_no	product_name	price
1	1	셀렉스 웨이 프로틴	5400

- product table에서 1번 상품의 수량은 12개에서 2개 감소한 10개가 됨
- purchase_history에서 1번 회원의 1번 제품 누적 구매 금액은 1800원에서 2개 구매를 더 한 5400원이 됨

- Custom 6: 사용자로부터 주소를 입력 받고 해당 문자열을 포함하는 회원 또는 트레이너를 검색함. 이때 회원은 결제 금액 순, 트레이너는 급여순으로 보여줌





- 주소는 텍스트로 입력 받고 member에 대해 조회하는지 trainer에 대해 조회하는지는 comboBox를 통해 입력받음

```
DECLARE @P_TABLE VARCHAR(10);

SET @P_TABLE = 'member'; -- 사용자 입력

IF(@P_TABLE = 'member')
BEGIN
    SELECT * FROM member
    WHERE address like '%호암로%'--사용자 입력
    ORDER BY amount_of_payment DESC, member_no ASC
END
ELSE IF(@P_TABLE = 'trainer')
BEGIN
    SELECT * FROM trainer
    WHERE address like '%호암로%'--사용자 입력
    ORDER BY salary DESC, trainer_no ASC
END
```