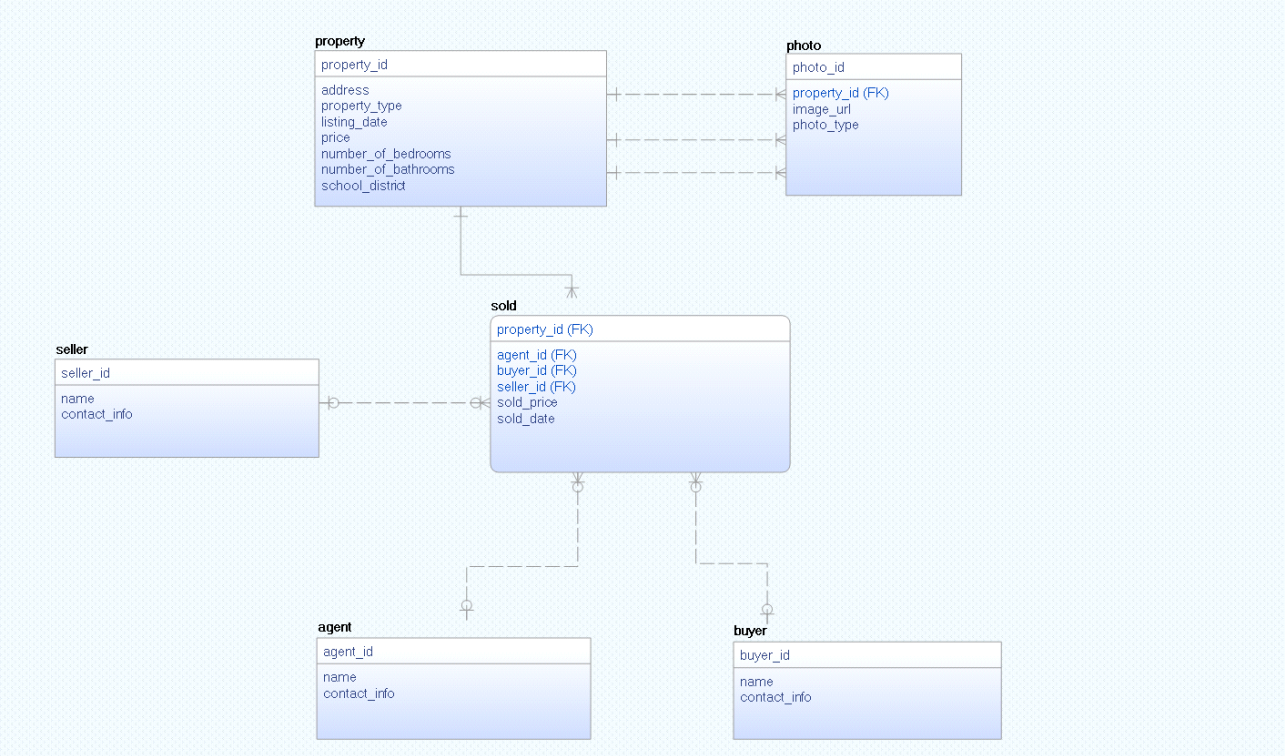
데이터베이스 시스템 프로젝트2 보고서

20190741 김지수

1. BCNF Decomposition



단계 1: BCNF 따져보기

Schema가 BCNF를 만족하는지 확인하려면, 해당 schema의 모든 함수 종속 α → β에 대하여, ① α가 superkey가 아니며, ② trivial하지 않은 α → β가 존재하는지 확인한다. 그러기 위해서는 해당 스키마에 도대체 어떤 함수 종속이 존재하는지를 부동산 시장의 도메인 지식을 활용해서 따져 보아야 한다.

따라서 각 스키마에서 우리에게 필요한 함수 종속에 관해서 생각해보자. 기존의 Relational model에는 property, photo, sold, seller, agent, buyer가 있다.

우선 가장 고려하기 단순한 seller, agent, buyer 개체에서는 해당 속성의 주 키에서 나머지 속성으로 가는 함수 종속만 존재하는 것이 자명하다. Multi-valued dependencies를 다루는 것이 아니기 때문에 편의상 시스템에는 하나의 연락처만 기록한다고 설정하고 과제의 편의성을 위해서 NULL 값이 가능하다고 하였다. name → contact_info나 name → id 로 가는 함수 종속은 성립하지 않는 것을 알 수 있다. 동명이인이 있을 수 있기 때문이다. 따라서 위의 3가지 개체집합은 BCNF 조건을 만족한다.

다음으로 시스템에서 이미 판매된 부동산 정보를 나타내는 sold에서 발생하는 함수종속을 고려한다. 해당 개체의 주키로는 외래키인 부동산 id와 등록날짜를 설정했다. sold는 property에 종속되는 weak entity set이다. 이미 데이터베이스 상에 판매로 등록된 부동산 매물만 판매기록을 남기도록 구현한 것이다. 만약 부동산을 삭제하고 싶다면 똑같은 property_id와 listing_date 속성을 사용해서 property와 sold에서 삭제하면 된다. 여기서 필요하다고 생각될 수 있는 함수 종속을 생각해 보았다.

agent_id, seller_id, buyer_id → sold_price, sold_date 라고 생각할 수도 있다. 하지만, 부동산 중개인과 구매자 판매자가 동일해도 해당 거래는 동일하지 않을 수 있다. 따라서 해당 함수 종속은 성립하지 않

는다. 그렇다면 property_id → sold_price, sold_date는 어떨까? 이 또한 동일한 부동산 id라고 할지라도 해당 부동산이 여러 차례 거래되었을 수 있기 때문에 해당 함수 종속이 반드시 성립하지는 않는 것을 알 수 있다. 그렇다면 agent_id, seller_id, buyer_id, listing_date → (something) 으로 가는 함수 종속이 가능할까? 이 또한 중개인과 구매자, 판매자가 같은 날짜에 등록되었지만 여러 개의 서로 다른 매물을 거래하는 경우도 존재할 수 있으므로 아니다. property_id, agent_id, seller_id, buyer_id → (something) 으로 가는 것 또한 동일한 매물에 대해서 같은 중개인과 구매자, 판매자가 서로 다른 날짜에 재거래하는 경우가 존재할 수 있으므로 아니다. 정리하자면, 존재하는 부동산 거래는 property_id, listing_date → (something) 으로 가는 함수 종속이 존재하는데 이들이 주 키(슈퍼키의 일종)이므로 BCNF를 만족한다.

참고사항으로 이전 프로젝트 보고서에서 같은 날짜에 같은 부동산이 판매를 위해 등록되는 경우 (해당 날짜에 등록 및 판매가 되고 거래를 취소하기 위해 재등록되기 보다는 그냥 거래를 취소되는 것으로 간주하였음)가 없다고 설정했기 때문에 이들은 주키이다. property_id와 listing_date가 동일한 entry는 해당 개체의 나머지 속성인 address, price, size, property_type, number_of_bedrooms, number_of_bathrooms, school_district가 동일하다. 같은 부동산 매물개체여야 하기 때문이다. 이를 통해 동일한 부동산이 여러 번 거래되는 경우에도 property_id는 동일하나 listing_date를 사용하여 구별할 수 있다. 물론 이것이 현재 판매중인지는 거래 장부를 기록하는 sold 개체를 통해 확인할 수 있다. 이는 추가적인 조인 연산이 필요하지 않도록 하여 경제적이다.

다음으로 property를 고려하였다. 여기서 property_id가 동일하면 price는 아니어도 address, size, property_type, number_of_bedrooms, number_of_bathrooms, school_district가 동일한 것을 확인할 수 있다. 따라서 이를 BCNF 분해 방법을 활용하여 분해할 것을 고려하였다. 그렇게 하는 경우 property_info (property_id, address, size, property_type, number_of_bedrooms, number_of_bathrooms, school_district) 와 property (property_id, listing_date, price)가 생성이 되는 것을 알 수 있는데, 이는 데이터베이스 시스템에서 쿼리를 사용하여 조회할 때 너무 많은 조인 연산을 필요로 할 것이라고 판단하여 다른 방법을 고안하였다.

해당 방법은 데이터베이스를 설계할 때 동일한 부동산이 여러 번 거래되면, 서로 다른 property_id를 사용하는 것이다. 기존에 방법에서는 동일 부동산 매물이 여러 번 거래되면, 동일한 property_id를 사용하되 listing_date를 주 키 집합에 포함시켜서 관리하였으나 이러한 방식보다는 이를 property_id 하나로 구별할 수 있도록 설계하는 것이 더 적합하다고 판단하였다. 이를 통해서 property를 참조하던 다른 개체 집합의 구성요소들도 보다 가벼워질 수 있었다. 추가적으로 프로젝트 1에서 property를 설계할 때 사용했던 size 속성은 본 프로젝트에서는 요구되지 않아 삭제하였다.

마지막으로 photo를 고려하였다. 여기서 image_url이 동일하면 다른 속성으로 가는 함수 종속이 존재한다. 하지만, image_url은 사진 각각을 모두 구별하는 슈퍼키이기 때문에 이는 BCNF를 위반하지 않는다. 다만 url을 주 키로 사용하는 것에는 많은 어려움이 있기 때문에 주키로 photo_id를 사용한 것이다. 또 다른 가능성으로는 property_id→ (something) 으로 가는 함수 종속이 존재한다고 생각할 수도 있다. 왜냐하면 id는 property를 구분하는 주 키이기 때문이다. 그러나 같은 부동산에 여러 사진이 존재할 수 있기 때문에 이는 사진을 구별하지는 않아 image_url이나 사진 종류로 가는 함수 종속이 존재하지 않는다. 이러한 이유로 photo 에는 슈퍼키 → (something) 으로 가는 함수 종속만 존재하여 BCNF를 만족한다.

단계 2: 최종 모델

1. property

여기서 address는 세부적인 주소까지 기록되어 있는 주소를 말하는데, 이 또한 슈퍼키이다. 하지만 긴 문자열을 데이터베이스 관리 상의 이유로 설계자가 주 키로 사용하지 않기로 결정하였다. 따라서 address → (something)으로 향하는 함수 종속은 BCNF를 만족한다. 이 외에는 만족해야할 함수종속이 존재하지 않는다. 왜냐하면 주소나 property_id 외의 속성은 현실 부동산 세계에서 다른 속성에 제한을 가하는 요소가 아니라 해당 부동산 매물의 단순한 특성이기 때문에 함수종속이 존재하지 않는다.

2. seller, buyer, agent

해당 개체 집합은 위에서 설명한 것과 동일하며 변화된 점이 존재하지 않는다. 한편 contact_info는 NULL 값이 존재할 수 있기 때문에 해당 속성으로부터 시작되는 함수 종속은 존재하지 않는다.

3. sold

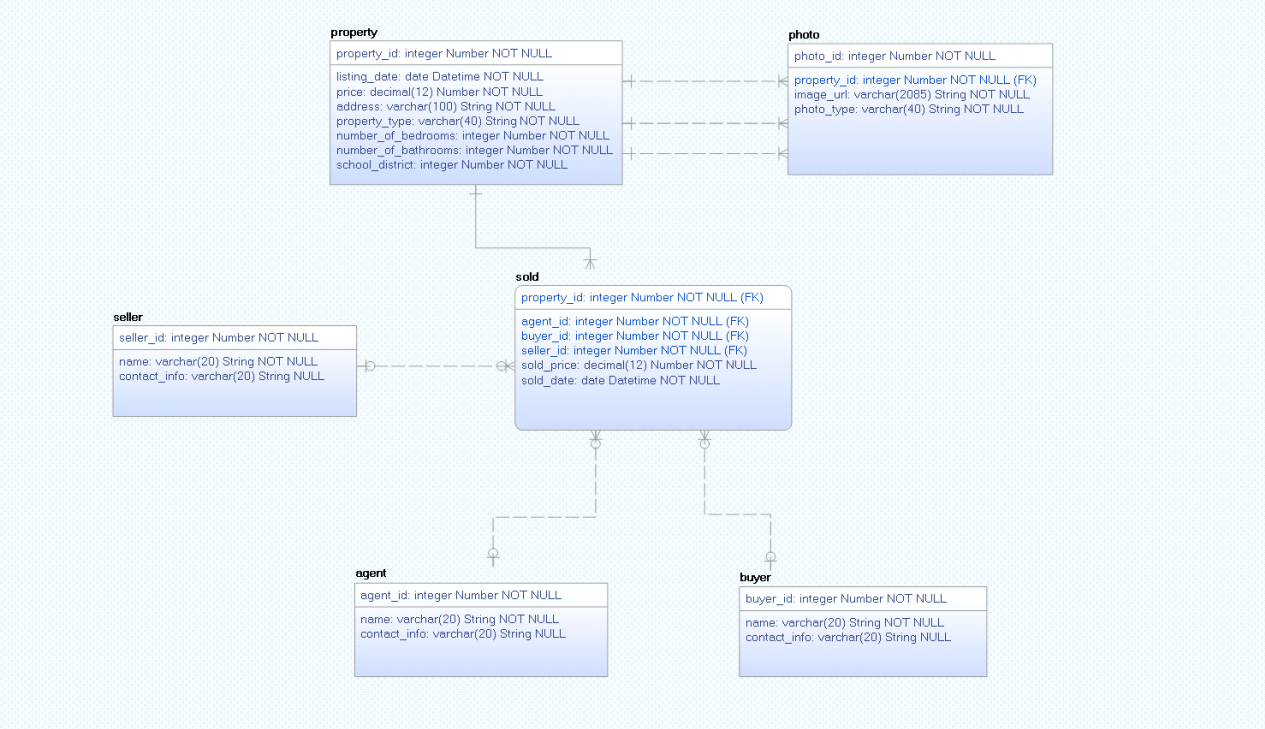
기존에 sold 개체에서 property 개체의 주 키 속성의 변화로 해당 부분이 변화된 점 이외에는 달라진 것이 없다. sold는 이미 등록된 부동산에 대한 정보가 기록되어야 한다는 점을 고려하여 property의 weak entity set으로 설계하였기 때문에 property_id만이 주 키로 사용되고 있다. 위에서 설명한 것과 같이 중개인과 구매자, 판매자가 동일해도 판매가격이나 판매날짜에 대한 함수 종속은 이들이 수차례 거래하였을 수 있기 때문에 존재할 수 없다. sold_price나 sold_date도 같은 날짜에 같은 가격에 거래되었다고 해서 다른 속성들을 특정할 수 있는 것이 아니기 때문에 알파가 슈퍼키가 아니면서 nontrivial한 함수 종속이 존재하지 않는 것을 알 수 있다.

4. photo

photo 속성 또한 기존의 설계에서 property 개체 집합의 주 키 변화로 listing_date가 삭제된 것 이외에 추가적인 변화는 존재하지 않는다. 나머지는 위에서 설명한 내용과 동일한 것으로 image_url 또한 슈퍼키이지만 긴 문자열을 가지는 속성이기에 주 키로 사용하지 않기로 결정하였다. 또한 property_id가 동일하여도 사진은 여러 장이며 photo_type또한 특정되지 않기 때문에 함수 종속이 성립하지 않는다. 이는 property_id, photo_type의 경우도 마찬가지로 같은 부동산의 같은 사진 타입이라 할지라도, 여러 장의 사진이 존재할 수 있다. 따라서 photo는 BCNF 조건을 만족한다.

Left Blank

2. Physical Schema Diagram



<Erwin Physical model>

위의 모형은 Erwin 프로그램을 사용하여 생성한 Physical schema diagram을 첨부한 것이며 아래에서 각각에 대하여 설명을 작성하였다.

3.1. seller

seller	Nullability	Type
seller_id	Not NULL	INTEGER PRIMARY KEY
name	Not NULL	varchar(20)
contact_info	Possibly NULL	varchar(20)

3.2. buyer

buyer	Nullability	Type
buyer_id	Not NULL	INTEGER PRIMARY KEY
name	Not NULL	varchar(20)
contact_info	Possibly NULL	varchar(20)

3.3. agent

agent	Nullability	Type
agent_id	Not NULL	INTEGER PRIMARY KEY
name	Not NULL	varchar(20)
contact_info	Possibly NULL	varchar(20)

부동산 판매자, 구매자, 그리고 이들의 대리인인 agent에 대한 기본 정보를 생각하여 개체를 구성하였다. 요구되는 쿼리에는 필요하지 않으나 기본적으로 연락처 정보와 이름은 필요할 것이라고 생각하여 추가하였고 연락처의 경우에는 프로젝트의 편의성을 위해 NULL 값이 가능하도록 설정하였다.

3.4. property

property	Nullability	Type
property_id	Not NULL	INTEGER PRIMARY KEY
listing_date	Not NULL	date Datetime
address	Not NULL	varchar(100)
price	Not NULL	Decimal(12, 0)
property_type	Not NULL	varchar(40)
number_of_bedrooms	Not NULL	INTEGER
number_of_bathrooms	Not NULL	INTEGER
school_district	Not NULL	INTEGER

부동산 정보를 저장하는 property 개체에는 모든 부동산의 정보를 담고 있다. 만약 해당 부동산이 팔린 것인지 확인하기 위해서는 sold 개체에 있는지 여부를 확인하기 위해서 NOT IN 연산을 사용하면 간단하게 확인할 수 있다. 한편 기존 보고서에서 설계했던 Type에서 약간의 변동이 있었다. 일반적으로 부동산이 평균 판매되기 까지 걸린 시간을 나타낼 때에는 시, 분, 초 단위가 아니라 일수를 기준으로 한다는 점을 고려하여 본 프로젝트의 설계에서도 listing_date와 sold 객체의 sold_date를 날짜를 기록하는 date로 변경하였다. 추가적으로 price은 기존 INTEGER가 프로그래밍 언어에서 통용되는 정수형 타입의 범위를 나타낼 수 있다고 생각하였고 매물의 가격이 이를 뛰어 넘을 수 있음을 고려하여 Decimal(12)로 변경하였다. address의 경우에도 기존 40에서 varchar을 사용하므로 긴 주소지를 고려하여 100으로 변경하였다. 프로젝트에 꼭 사용되는 부동산 매물에 대한 속성만 설계하여 모두 Not Null로 설정하였다.

3.5. sold

sold	Nullability	Type
property_id	Not NULL	INTEGER PRIMARY KEY, FK
agent_id	Not NULL	INTEGER, FK
buyer_id	Not NULL	INTEGER, FK
seller_id	Not NULL	INTEGER, FK
sold_price	Not NULL	Decimal(12, 0)
sold_date	Not NULL	date Datetime

sold 개체는 property 개체에 종속적이다. 즉 약한 개체 집합으로, sold는 판매되는 해당 property가 없으면 존재하지 않는다. 다시 말해서, 부동산이 기존에 이미 등록되어야지 판매될 수 있는 것이다. 따라서 primary key로는 강한 개체집합인 property의 primary key와 필요한 경우 sold의 속성을 primary key로 갖는다. 기존의 설계와 달라진 것은 동일한 부동산이 여러 번 거래되면 property_id가 변한다는 것이다. 이는 또한 부동산이라는 재화의 속성을 반영한 것이다. 실제로, 부동산은 내부 인테리어에 따라서 가벽을 설치하여 방의 개수나 다른 속성이 상황에 따라서 변할 수 있기에 하나의 매물에 대해서 고정된 property를 정하기보다는 하나의 매물이라도 거래의 시점에 따라서 해당 속성을 기록한 것이다. 이것은 부동산이라는 재화가 일반 재화랑은 다르게 하나의 개체에 대해서 빈번하게 거래가 이루어지지 않는 점을 고려한 설계이다.

추가로 sold 개체는 agent_id, seller_id, buyer_id를 foreign key로 갖는다. 이는 join 연산을 통해서 구매자, 판매자, 중개인(agent)이 판매된 부동산 매물 중 어떤 것과 관련이 있는지 확인하기 위함이다. 여기서 property의 price와 sold의 sold_price는 다른 것이다. 왜냐하면 판매자가 부동산을 등록한 가격과 구매자와 판매자가 합의하여 최종적으로 거래된 가격은 다를 수 있기 때문이다. 실제로 agent의 실적으로 고려하는 것에는 최종 판매된 가격이 사용된다. 한편, 구입할 수 있는 부동산의 가격을 고려할 때는

property의 price가 고려되는 것이다. sold 개체에 있는 sold_date와 property에 있는 listing_date 속성을 활용해서 해당 부동산 매물이 판매되기까지 소요된 기간을 구하였다.

sold 개체의 Type에서는 sold_price를 property의 price와 동일하게 Decimal(12)로 변경하였으며 sold_date도 마찬가지로 property의 listing_date와 동일하게 date Datetime으로 변경하였다. 거래가 이루어지는 부동산, 구매자, 판매자, 대리인, 가격, 판매일 모두 요구되는 쿼리에서 반드시 필요한 정보이므로 모두 Not NULL로 설정하였다.

3.6 photo

photo 개체는 부동산의 사진 정보를 담는 개체로, photo_type은 interior_photo, exterior_photo, floor_plan 등의 값을 갖는다. photo는 photo_id를 primary key로 갖도록 설정하였다. 해당 이유는 사진의 장수에는 제한이 없기 때문에, 동일 부동산의 동일 인테리어 사진에 대해서도 이미지가 여러 장이 존재할 수 있기 때문이다. URL의 경우에는 언제나 상황에 따라서 변경될 수 있기 때문에 primary key로 사용하는 것을 고려하지 않았다. 추가적으로 해당 사진이 어느 부동산과 관련이 있는 사진인지를 나타내기 위해서 property_id를 foreign key로 갖는다.

photo	Nullability	Type
photo_id	Not NULL	INTEGER PRIMARY KEY
image_url	Not NULL	varchar(2085)
photo_type	Not NULL	varchar(40)
property_id	Not NULL	INTEGER, FK

photo가 property의 primary key를 foreign key로 가지기 때문에 해당 부동산의 사진은 property와 photo의 join을 통해서 구할 수 있다. 또한 “데이터 베이스에서 가장 비싼” 이라는 조건이므로 property에는 판매된 부동산이나 아직 미판매된 부동산 데이터가 모두 작성되어 있어 해당 쿼리를 쉽게 수행할 수 있다.

여기서 image_url은 사진이 저장되어 있는 인터넷 상의 주소를 지칭하는데, 일반적으로 통용되는 url의 길이에 여유를 두어 2085로 설정했으며, 낭비되는 것을 방지하기 위해 varchar 타입으로 설정하였다. 사진의 id, 사진의 주소, 타입, 그리고 어느 부동산의 사진인지를 나타내줄 property_id는 모두 요구되는 쿼리 수행에서 필요한 정보이기에 Not Null로 설정하였다.

Left Blank

3. About ODBC

3.1 CRUD FILE

Query Processing에 앞서서 데이터베이스에 테이블과 값들을 입력하는 작업이 필요하다. 사용자가 자신이 원하는 Query를 프로그램에 입력하기 전에 프로그램 코드 상으로 텍스트 파일 처리를 하여 ODBC를 통해 연결된 데이터베이스 상에 테이블을 생성하고 알맞은 데이터를 입력한다. 여기서 데이터 입력에 딱 한 가지 가정이 이루어진다. 사용자가 사진을 입력할 때 부동산의 타입에 맞게 사진의 개수를 정확하게 입력한다고 본 프로젝트에서는 가정한다. 마지막으로 가장 처음에 메뉴에서 QUIT을 입력하면 생성한 테이블을 모두 제거하고 프로그램이 종료되는 방식으로 설계하였다. 아래는 CRUD FILE의 일부를 예시로 첨부하였다.

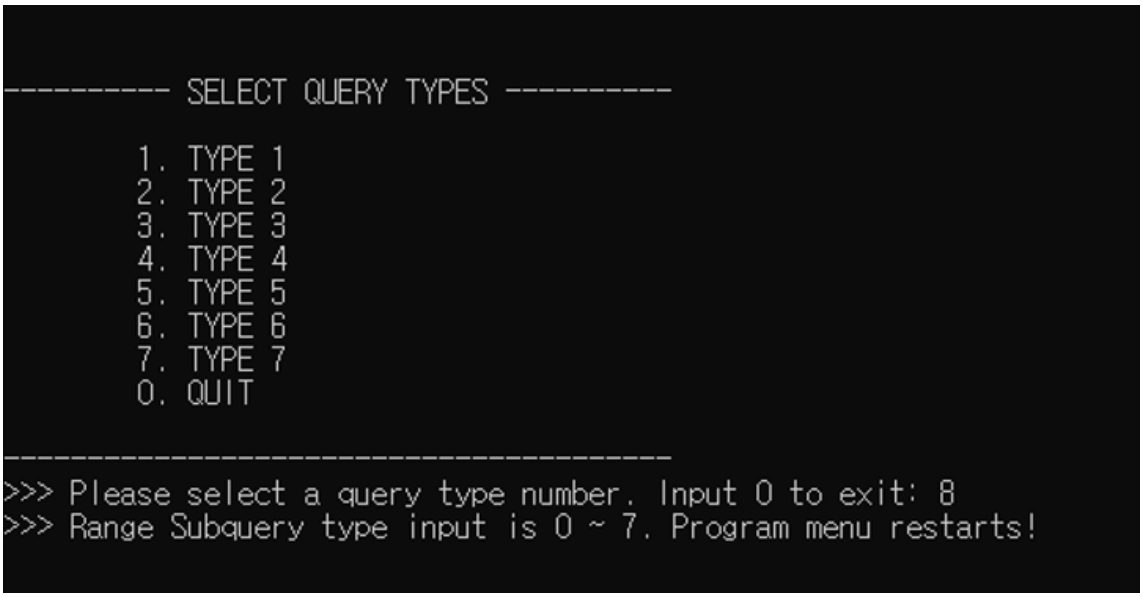
```
CREATE TABLE property (property_id INTEGER NOT NULL, listing_date DATE NOT NULL, price DECIMAL(12, 2) NOT NULL, address VARCHAR(100) NOT NULL, property_type VARCHAR(40) NOT NULL, number_of_bedrooms INTEGER NOT NULL, number_of_bathrooms INTEGER NOT NULL, school_district INTEGER NOT NULL, PRIMARY KEY (property_id));
CREATE TABLE photo (photo_id INTEGER NOT NULL, property_id INTEGER NOT NULL, image_url VARCHAR(2085) NOT NULL, photo_type VARCHAR(40) NOT NULL, PRIMARY KEY (photo_id), FOREIGN KEY (property_id) REFERENCES property(property_id));
CREATE TABLE seller (seller_id INTEGER NOT NULL, name VARCHAR(20) NOT NULL, contact_info VARCHAR(20) NULL, PRIMARY KEY (seller_id));
CREATE TABLE agent (agent_id INTEGER NOT NULL, name VARCHAR(20) NOT NULL, contact_info VARCHAR(20) NULL, PRIMARY KEY (agent_id));
CREATE TABLE buyer (buyer_id INTEGER NOT NULL, name VARCHAR(20) NOT NULL, contact_info VARCHAR(20) NULL, PRIMARY KEY (buyer_id));
CREATE TABLE sold (property_id INTEGER NOT NULL, agent_id INTEGER NOT NULL, buyer_id INTEGER NOT NULL, seller_id INTEGER NOT NULL, sold_price DECIMAL(12, 2) NOT NULL, sold_date DATE NOT NULL, PRIMARY KEY (property_id, agent_id, buyer_id, seller_id), FOREIGN KEY (property_id) REFERENCES property(property_id), FOREIGN KEY (agent_id) REFERENCES agent(agent_id), FOREIGN KEY (buyer_id) REFERENCES buyer(buyer_id), FOREIGN KEY (seller_id) REFERENCES seller(seller_id));
/*value 삽입 코드는 생략*/
$$$
drop table sold;
drop table photo;
drop table seller;
drop table buyer;
drop table agent;
drop table property;
```

example.cpp 코드를 살펴보면 하나의 CRUD 파일에서 $$$\n을 통해 데이터베이스의 테이블 및 데이터 값을 입력받고 수행코드의 실행이 완료된 후 마지막에 테이블을 삭제하는 것을 알 수 있다. 따라서 위의 코드는 해당 형식에 맞춰 설계한 테이블을 서버를 통해 생성한 것이다. 다만 drop table을 할 때 참조하는 FK를 고려하여 삭제하는 순서에 유의하여 CRUD 파일을 작성하였다. 다시 말해서, sold는 다양한 테이

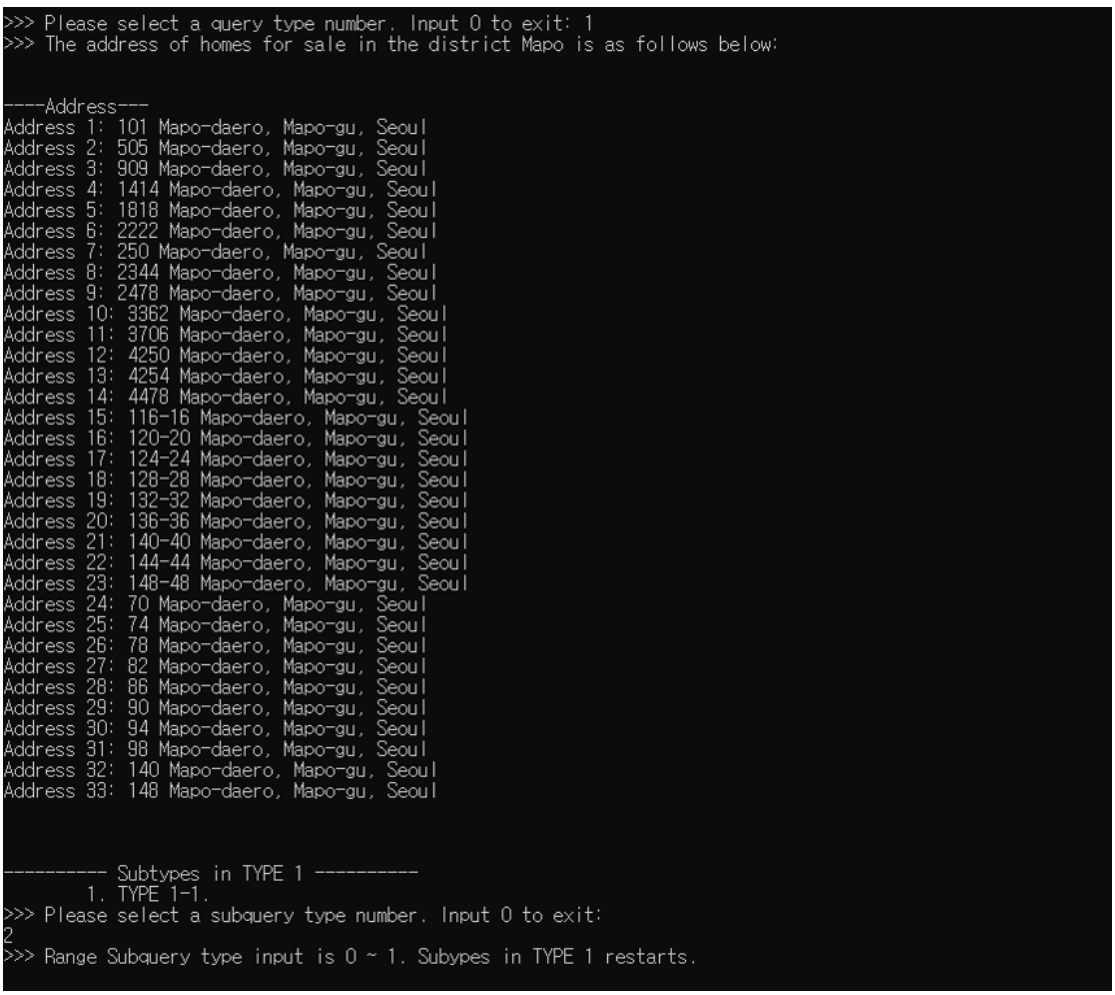
블에서 참조하는 FK가 존재하여 가장 먼저 삭제하였다.

3.2 Query Processing

먼저 연결이 완료된 후의 메인화면이다. 아래의 사진결과처럼 범위 밖에 수를 입력하는 경우에 프로그램이 이를 공지하고 다시 실행되는 것을 알 수 있다.



(Type 1)



위의 사진처럼 타입 1이 실행되면 자동으로 (1)에 해당하는 쿼리가 수행되고 해당하는 결과 값들이 출력되는 것을 확인할 수 있다. Mapo-gu라는 주소의 substring과 매치되는 것중 sold에 없는(not in)것을 조회하여 출력한 것으로 마포구에서 현재 판매중인 매물의 주소가 위와 같이 출력되었다. SQL 쿼리문은 아래와 같고 서브 타입에서도 범위를 잘못입력하면, 이를 공지하고 서브 타입 부분만 다시 실행시킨다.

SELECT address FROM property WHERE address LIKE '%%Mapo-gu%%' AND property_id NOT IN (SELECT property_id FROM sold);
%%기호는 실제로는 snprintf 함수를 사용하여 작성하였기에 문자열로 만들고 이를 전송하기 위해 위와 같이 작성하였다.

(TYPE 1-1)

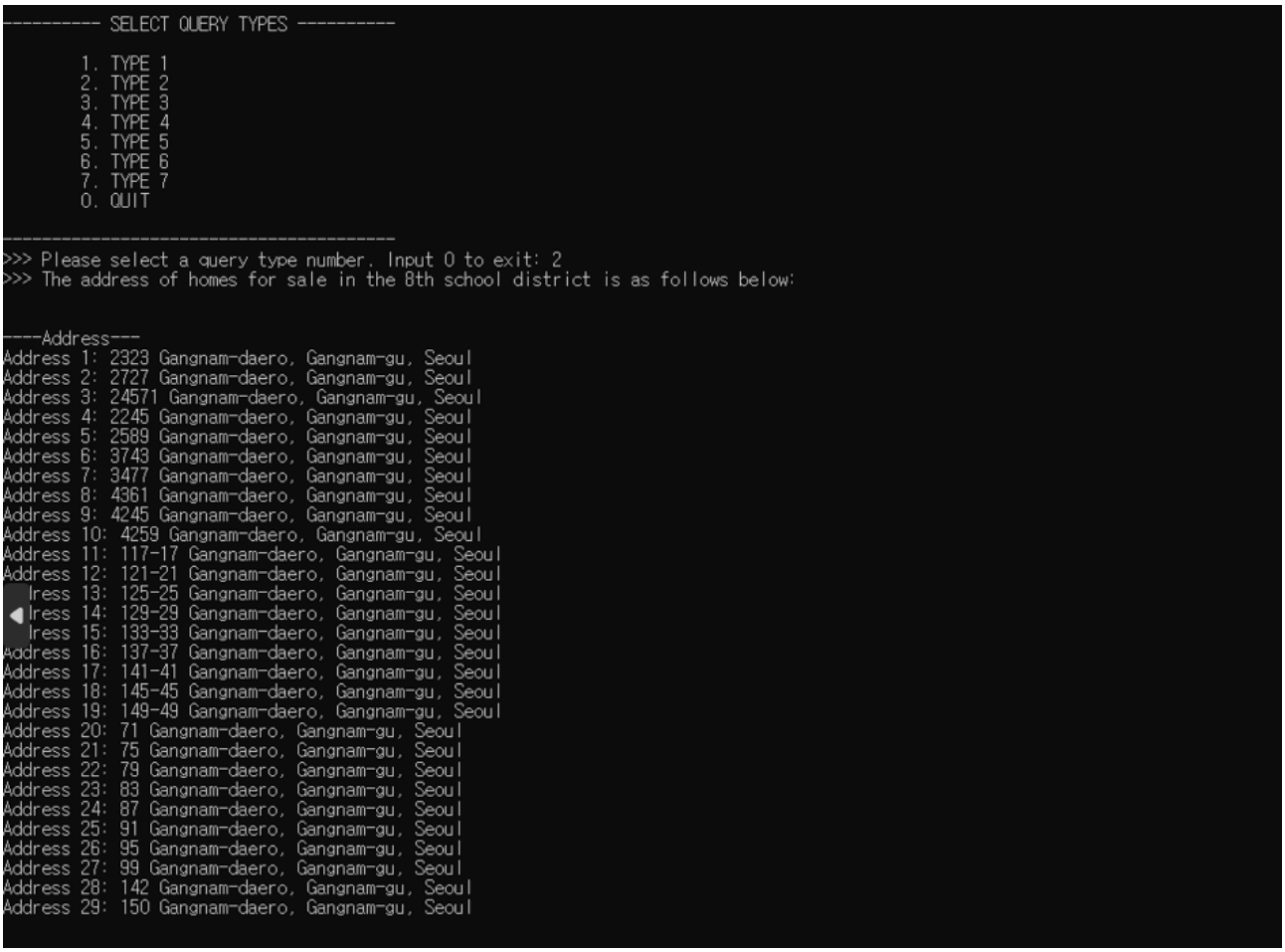
```
---------- Subtypes in TYPE 1 ----------
       1. TYPE 1-1.
>>> Please select a subquery type number. Input 0 to exit:
1
>>> The address of homes for sale in the district Mapo with price between ₩1,000,000,000 and ₩1,500,000,000 is as follows below:


----Address---
Address 1: 101 Mapo-daero, Mapo-gu, Seoul
Address 2: 2344 Mapo-daero, Mapo-gu, Seoul
Address 3: 2478 Mapo-daero, Mapo-gu, Seoul
Address 4: 3362 Mapo-daero, Mapo-gu, Seoul
Address 5: 3706 Mapo-daero, Mapo-gu, Seoul
Address 6: 4250 Mapo-daero, Mapo-gu, Seoul
Address 7: 4254 Mapo-daero, Mapo-gu, Seoul
Address 8: 4478 Mapo-daero, Mapo-gu, Seoul
Address 9: 124-24 Mapo-daero, Mapo-gu, Seoul
Address 10: 128-28 Mapo-daero, Mapo-gu, Seoul
Address 11: 132-32 Mapo-daero, Mapo-gu, Seoul
Address 12: 136-36 Mapo-daero, Mapo-gu, Seoul
Address 13: 140-40 Mapo-daero, Mapo-gu, Seoul
Address 14: 144-44 Mapo-daero, Mapo-gu, Seoul
Address 15: 148-48 Mapo-daero, Mapo-gu, Seoul
Address 16: 74 Mapo-daero, Mapo-gu, Seoul
Address 17: 78 Mapo-daero, Mapo-gu, Seoul
Address 18: 82 Mapo-daero, Mapo-gu, Seoul
Address 19: 86 Mapo-daero, Mapo-gu, Seoul
Address 20: 90 Mapo-daero, Mapo-gu, Seoul
Address 21: 94 Mapo-daero, Mapo-gu, Seoul
Address 22: 98 Mapo-daero, Mapo-gu, Seoul
Address 23: 148 Mapo-daero, Mapo-gu, Seoul
```

서브타입인 1-1이 실행되면 아래와 같이 마포구에서 현재 판매중이면서 가격 범위가 제시된 범위의 매물들이 차례대로 출력된다. Address 뒤의 넘버는 매물 번호를 나타낸다. ODBC SQL 쿼리는 아래와 같다. 자세한 것은 코드에 설명을 덧붙였다.

SELECT address FROM property WHERE address LIKE '%%Mapo-gu%%' AND property_id NOT IN (SELECT property_id FROM sold) AND price BETWEEN 1000000000 AND 1500000000;
%%기호는 실제로는 snprintf 함수를 사용하여 작성하였기에 문자열로 만들고 이를 전송하기 위해 위와 같이 작성하였다.

(TYPE 2)



Type 2를 실행하였을 때 기본적으로 실행된 (3)번 항목의 결과는 위의 사진과 같다. 8학군의 데이터로 강남구만 데이터를 만들었기 때문에 위와 같이 나왔다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

snprintf(query1, sizeof(query1), "SELECT address FROM property WHERE school_district = 8 AND property_id NOT IN (SELECT property_id FROM sold);");

(TYPE 2-1)

Type 2-1를 실행하였을 때 실행된 (4)번 항목의 결과는 다음 페이지의 사진과 같다. 판매중인 8학군의 매물 데이터로 강남구만 데이터를 만들었기 때문에 위와 같이 나온 것에서 조건을 만족하는 매물이 실제로 없었고 따라서 아래의 사진과 같이 나온 것이다. 실제 사용된 ODBC SQL 쿼리 코드는 아래와 같다.

snprintf(query2, sizeof(query2), "SELECT address FROM property WHERE school_district = 8 AND property_id NOT IN (SELECT property_id FROM sold) AND number_of_bedrooms >= 4 AND number_of_bathrooms = 2;");

```
---------- Subtypes in TYPE 2 ----------
       1. TYPE 2-1.
>>> Please select a subquery type number. Input 0 to exit:
1
>>> The address of homes for sale in the 8th school district with 4 or more bedrooms and 2 bathrooms is as follows below:


----Address---
No homes for sale in the 8th school district with 4 or more bedrooms and 2 bathrooms.
```

(TYPE 3)

Type 3를 실행하였을 때 기본적으로 실행된 (5)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(query, sizeof(query),
    "SELECT a.name "
    "FROM sold s "
    "JOIN agent a ON s.agent_id = a.agent_id "
    "WHERE YEAR(s.sold_date) = 2022 "
    "GROUP BY a.agent_id, a.name "
    "ORDER BY SUM(s.sold_price) DESC "
    "LIMIT 1;");
```

```
---------- SELECT QUERY TYPES ----------

       1. TYPE 1
       2. TYPE 2
       3. TYPE 3
       4. TYPE 4
       5. TYPE 5
       6. TYPE 6
       7. TYPE 7
       0. QUIT

----------------------------------------
>>> Please select a query type number. Input 0 to exit: 3
>>> The name of the agent who has sold the most properties in the year 2022 by total won value is as follows below:
Agent Name: Arnaldo Rivera
```

(TYPE 3-1)

Type 3-1를 실행하였을 때 기본적으로 실행된 (6)번 항목의 결과는 다음 페이지의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(query3_1, sizeof(query3_1),
    "SELECT a.agent_id, a.name, SUM(s.sold_price) as total_sold "
    "FROM sold s "
    "JOIN agent a ON s.agent_id = a.agent_id "
    "WHERE YEAR(s.sold_date) = 2023 "
    "GROUP BY a.agent_id, a.name "
    "ORDER BY total_sold DESC "
    "LIMIT %d;", k);
```

```
---------- Subtypes in TYPE 3 ----------
        1. TYPE 3-1.
        1. TYPE 3-2.
>>> Please select a subquery type number. Input 0 to exit:
1
>>> Enter the number(k) of top agents to retrieve:
4


--- Top 4 agents by total won value in 2023 ---
Rank 1: Agent ID: 10007, Name: James Suckling, Total Sold: 12300000000
Rank 2: Agent ID: 10005, Name: Robert Parker, Total Sold: 6650000000
Rank 3: Agent ID: 10008, Name: Krug, Total Sold: 6170000000
Rank 4: Agent ID: 10001, Name: Arnaldo Rivera, Total Sold: 5450000000
```

(TYPE 3-2)

Type 3-2를 실행하였을 때 기본적으로 실행된 (7)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(count_query, sizeof(count_query),
    "SELECT FLOOR(COUNT(*) * 0.1) AS limit_value FROM agent;");
```

```
snprintf(query3_2, sizeof(query3_2),
    "SELECT a.agent_id, a.name, COALESCE(SUM(s.sold_price), 0) as total_sold "
    "FROM agent a "
    "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2021 "
    "GROUP BY a.agent_id, a.name "
    "ORDER BY total_sold ASC "
    "LIMIT %d;", limit_value);
```

```
---------- Subtypes in TYPE 3 ----------
        1. TYPE 3-1.
        2. TYPE 3-2.
>>> Please select a subquery type number. Input 0 to exit:
2


--- Bottom 10% agents by total won value in 2021 ---
Agent ID: 10010, Name: Crystal, Total Sold: 1300000000
```

(TYPE 4)
Type 4를 실행하였을 때 기본적으로 실행된 (8)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(query, sizeof(query),
    "SELECT a.agent_id, a.name, "
    "IFNULL(AVG(s.sold_price), '-') AS avg_selling_price, "
    "IFNULL(AVG(DATEDIFF(s.sold_date, p.listing_date)), '-') AS avg_market_time "
    "FROM agent a "
    "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2022 "
    "LEFT JOIN property p ON s.property_id = p.property_id "
    "GROUP BY a.agent_id, a.name;");
```

```
---------- SELECT QUERY TYPES ----------

       1. TYPE 1
       2. TYPE 2
       3. TYPE 3
       4. TYPE 4
       5. TYPE 5
       6. TYPE 6
       7. TYPE 7
       0. QUIT

---------------------------------------
>>> Please select a query type number. Input 0 to exit: 4
>>> The average selling price and average market time for each agent in 2022 are as follows below:


Agent ID   Name               Avg Selling Price    Won Avg Market Time      Days
10001      Arnaldo Rivera     2700000000.0000      Won 191.5000             Days
10002      Olive              1225000000.0000      Won 174.0000             Days
10003      Mike               1750000000.0000      Won 170.0000             Days
10004      Vogue              1125000000.0000      Won 168.0000             Days
10005      Robert Parker      1675000000.0000      Won 166.0000             Days
10006      Jancis Robinson    1075000000.0000      Won 165.5000             Days
10007      James Suckling     1750000000.0000      Won 179.0000             Days
10008      Krug               1225000000.0000      Won 178.5000             Days
10009      Margaux            1775000000.0000      Won 165.5000             Days
10010      Crystal            1300000000.0000      Won 312.0000             Days
```

실제로 데이터베이스 상에 등록된 모든 agent에 대해서 이를 계산하도록 설정하였다. 값이 존재하지 않는다면 - 로 표기될 것이다.

(TYPE 4-1)
Type 4-1를 실행하였을 때 기본적으로 실행된 (9)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(query4_1, sizeof(query4_1),
    "SELECT a.agent_id, a.name, "
    "IFNULL(MAX(s.sold_price), '-') AS max_selling_price "
    "FROM agent a "
    "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2023 "
    "GROUP BY a.agent_id, a.name;");
```

```
---------- Subtypes in TYPE 4 ----------
        1. TYPE 4-1
        2. TYPE 4-2
>>> Please select a subquery type number. Input 0 to exit:
1


--- Maximum selling price of properties sold in 2023 for each agent ---
Agent ID: 10001, Name: Arnaldo Rivera, Max Selling Price: 3550000000 Won
Agent ID: 10002, Name: Olive, Max Selling Price: 1550000000 Won
Agent ID: 10003, Name: Mike, Max Selling Price: 1900000000 Won
Agent ID: 10004, Name: Vogue, Max Selling Price: 1400000000 Won
Agent ID: 10005, Name: Robert Parker, Max Selling Price: 2200000000 Won
Agent ID: 10006, Name: Jancis Robinson, Max Selling Price: 1500000000 Won
Agent ID: 10007, Name: James Suckling, Max Selling Price: 4500000000 Won
Agent ID: 10008, Name: Krug, Max Selling Price: 3200000000 Won
Agent ID: 10009, Name: Margaux, Max Selling Price: 2050000000 Won
Agent ID: 10010, Name: Crystal, Max Selling Price: 1300000000 Won
```

(TYPE 4-2)

Type 4-2를 실행하였을 때 기본적으로 실행된 (10)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
snprintf(query4_2, sizeof(query4_2),
    "SELECT a.agent_id, a.name, "
    "IFNULL(MAX(DATEDIFF(s.sold_date, p.listing_date)), '-') AS longest_market_time "
    "FROM agent a "
    "LEFT JOIN sold s ON a.agent_id = s.agent_id "
    "LEFT JOIN property p ON s.property_id = p.property_id "
    "GROUP BY a.agent_id, a.name;");
```

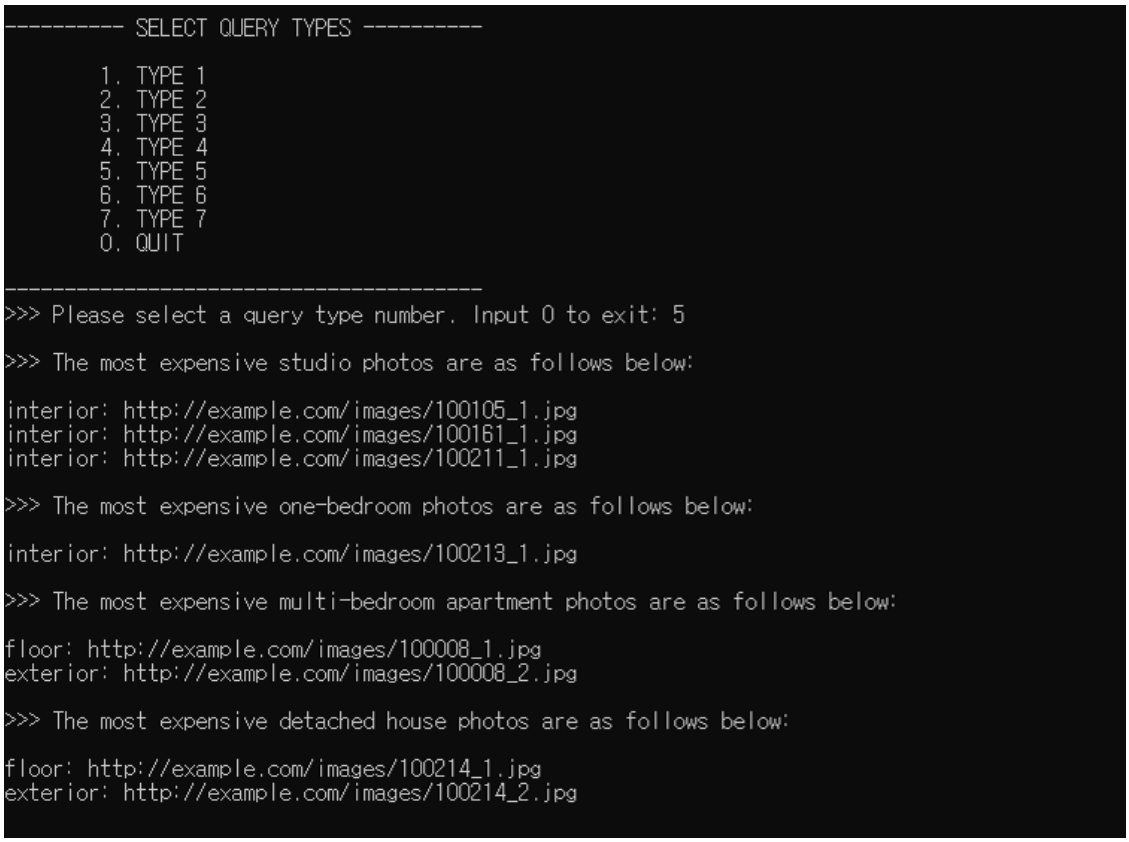
```
---------- Subtypes in TYPE 4 ----------
        1. TYPE 4-1
        2. TYPE 4-2
>>> Please select a subquery type number. Input 0 to exit:
2


--- Longest time the property was on the market for each agent ---
Agent ID: 10001, Name: Arnaldo Rivera, Longest Market Time: 344 Days
Agent ID: 10002, Name: Olive, Longest Market Time: 287 Days
Agent ID: 10003, Name: Mike, Longest Market Time: 246 Days
Agent ID: 10004, Name: Vogue, Longest Market Time: 209 Days
Agent ID: 10005, Name: Robert Parker, Longest Market Time: 487 Days
Agent ID: 10006, Name: Jancis Robinson, Longest Market Time: 427 Days
Agent ID: 10007, Name: James Suckling, Longest Market Time: 610 Days
Agent ID: 10008, Name: Krug, Longest Market Time: 668 Days
Agent ID: 10009, Name: Margaux, Longest Market Time: 277 Days
Agent ID: 10010, Name: Crystal, Longest Market Time: 313 Days
```

(TYPE 5)
Type 5를 실행하였을 때 기본적으로 실행된 (11)번 항목의 결과는 아래의 사진과 같다. 실제 사용된 ODBC SQL 쿼리코드는 아래와 같다.

```
char room_types[4][20] = { "studio", "one-bedroom", "multi-bedroom", "detached" };
char room_descriptions[4][100] = { "most expensive studio", "most expensive one-bedroom",
"most expensive multi-bedroom apartment", "most expensive detached house" };

for (int i = 0; i < 4; i++) {
    snprintf(query, sizeof(query),
        "SELECT p.photo_type, p.image_url FROM photo p "
        "JOIN property pr ON p.property_id = pr.property_id "
        "WHERE pr.property_type = '%s' "
        "AND pr.price = (SELECT MAX(price) FROM property WHERE property_type = '%s');",
        room_types[i], room_types[i]);
        /*이하 생략*/
}
```



반복문을 이용해서 등록된 각 타입 별로 위의 ODBC SQL 쿼리를 수행하는 코드를 작성하고 이를 받아서 출력하였다. 실제로 출력된 결과는 가장 비싼 매물의 값이 동일한 경우가 포함될 수 있을 수 있기 때문에 사진 url이 studio나 one-bedroom (interior 사진이 최소 한 장이지만 한 장만 등록) 에 대해서도 여러개 나타날 수 있다. 이 경우 url에서 property_id를 기준으로 주소를 생성하였기에 확인은 가능하다. 추가로 매물에 대해서 사진이 여러 장인 경우에는 property_id_1, property_id_2 등의 방법으로 url의 세부주소에서 확인이 가능하다. 데이터를 입력할 때 기본적으로 타입 별로 주어진 사진의 장수제한은 사용자가 정확하게 조건에 따라 입력한다고 가정하였다.

(TYPE 6)
Type 6를 실행하였을 때 기본적으로 실행된 (12)번 항목의 결과는 아래의 사진과 같다. 최종적으로 값을 기록하는 쿼리 코드는 아래와 같다.
여기서 사용자가 입력한 부동산 id가 없거나 이미 판매된 경우, seller, buyer, agent의 이름이 데이터베이스 상에 존재하지 않는 없는 사람인 경우에 대해서 다시 입력을 하라고 설정했다. 우선 프로젝트의 편의성을 위해서 이들이 각 집합에서 주 키 속성은 아니지만 seller, buyer, agent에서 중복된 이름은 존재하지 않는다고 하였다.
다만 한 가지 유의할 점은 다른 속성에 대해서는 무결성을 검사하지만, sold_date에 대해서는 사용자가 부동산 등록일 이후의 날짜로 입력한다는 것을 가정하였다.

snprintf(query_insert, sizeof(query_insert),
"INSERT INTO sold (property_id, agent_id, buyer_id, seller_id, sold_price, sold_date)
VALUES (%d, %d, %d, %d, %lf, '%s');", property_id, agent_id, buyer_id, seller_id, sold_price, sold_date);

```
---------- SELECT QUERY TYPES ----------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        6. TYPE 6
        7. TYPE 7
        0. QUIT

----------------------------------------
>>> Please select a query type number. Input 0 to exit: 6
>>> Record the sale of a property.
Enter the property code (property_id): 123456
Enter the sold price: 100000000
Enter the buyer name: Canon
Enter the agent name: Robert Parker
Enter the seller name: Palmer
Enter the sold date (YYYY-MM-DD): 2024-06-12
Invalid property code or the property is already sold.
```

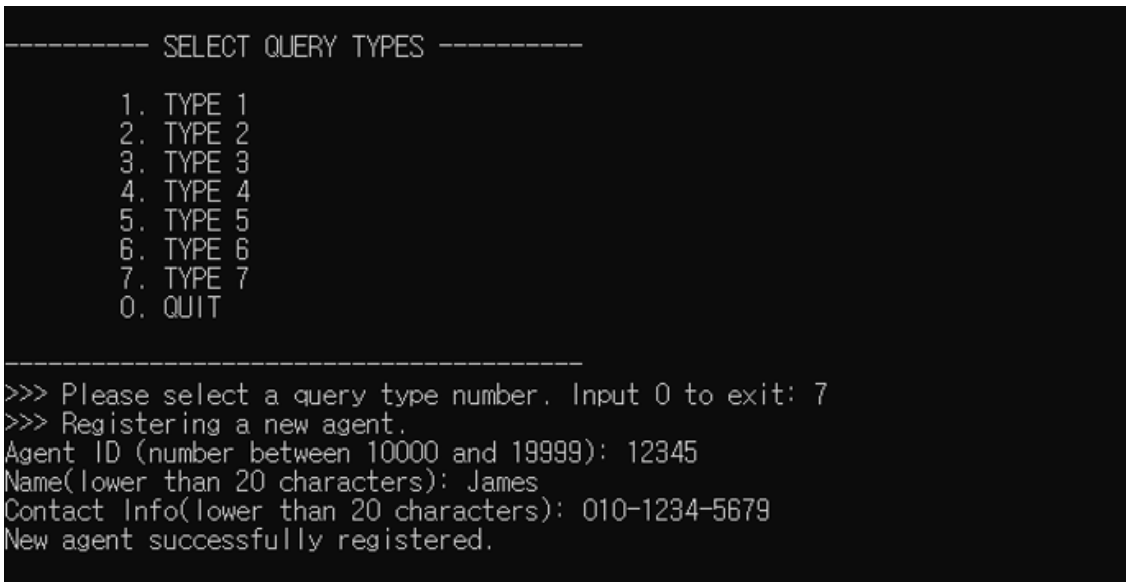
위의 사진과 같이 입력된 부동산이 실제로 데이터베이스 상에는 존재하지 않는 매물인 경우나 존재는 하나 이미 sold에 기록되어 판매가 완료된 데이터인 경우에는 데이터베이스 상으로 등록할 수 없기 때문에 이를 공지하고 다시 메인 메뉴로 돌아간다.

한편 다음 페이지의 사진과 같이 정상적으로 값들이 입력된 경우에는 데이터베이스에 등록하고 결과가 성공적으로 실행되었다는 문구를 출력하는 것을 확인할 수 있다.

(TYPE 7)
Type 7를 실행하였을 때 기본적으로 실행된 (13)번 항목의 결과는 아래의 사진과 같다. 최종적으로 값을 기록하는 쿼리 코드는 아래와 같다.

```
snprintf(query, sizeof(query),
    "INSERT INTO agent (agent_id, name, contact_info) VALUES (%d, '%s', '%s');",
    agent_id, name, contact_info);
```

이는 agent가 새로 자신을 데이터베이스에 등록하는 과정으로 아이디를 입력하는 과정이 필요하다. 여기서 아이디의 범위를 프로그램이 제시하고 중복이 되는 경우에는 다시 입력을 하도록 프로그램을 구성하였다. 다만, name과 contact_info에서 공백(스페이스)은 허용하지 않고 사용자가 이를 정상적으로 입력할 것을 가정하였다.



4. Code Description

아래는 최종적으로 수행된 프로젝트 코드에 주석을 덧붙인 설명입니다. 참고용으로 첨부하였습니다.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <cstring>
#include "mysql.h"

#pragma comment(lib, "libmysql.lib")

const char* host = "localhost"; // change if necessary
const char* user = "root";      // change if necessary
const char* pw = "@2997299k";         // change if necessary
const char* db = "project";

#define MAX_LEN 13000

int main(void) {
    MYSQL* connection = NULL;
    MYSQL conn;
    MYSQL_RES* sql_result;
    MYSQL_ROW sql_row;

    FILE* fp = fopen("CRUD.txt", "r");     // open CRUD file.
    if (fp == NULL) {
        fprintf(stderr, "Input CURD file open error!\n");
        exit(0);
    }

    char line[MAX_LEN];
    if (mysql_init(&conn) == NULL)
        printf("mysql_init() error!");
    // MySQL 서버에 연결합니다.
    connection = mysql_real_connect(&conn, host, user, pw, NULL, 3306, (const char*)NULL, 0);    // the first NULL can be replaced to an existing db
instance.
    if (connection == NULL) {
        printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
        return 1;
    }
    else {
        printf("Connection Succeed\n\n");
        // 프로젝트 데이터베이스를 선택합니다.
        if (mysql_select_db(&conn, "project")) {
            printf("%d ERROR : %s\n", mysql_errno(&conn), mysql_error(&conn));
            return 1;
        }
        // CRUD.txt 파일에서 읽은 라인들을 실행합니다.
        while (fgets(line, sizeof(line), fp) != NULL) {
            if (!strcmp(line, "$$$\n"))        // read lines from CRUD file, '$$$' separates CREATE / DELETE parts.
                break;
            mysql_query(connection, line);
        }
        // 메인 쿼리 유형 선택 루프
        while (1) {
            int input = 0;
            printf("\n\n---------- SELECT QUERY TYPES ----------\n\n");
            printf("\t1. TYPE 1\n");
            printf("\t2. TYPE 2\n");
            printf("\t3. TYPE 3\n");
            printf("\t4. TYPE 4\n");
            printf("\t5. TYPE 5\n");
            printf("\t6. TYPE 6\n");
            printf("\t7. TYPE 7\n");
            printf("\t0. QUIT\n");
            printf("\n----------------------------------------\n");


            printf(">>> Please select a query type number. Input 0 to exit: ");
            scanf("%d", &input);
            // 프로그램 종료
            if (input == 0) break;
```

```
        else if (input == 1) {
            // TYPE 1: 특정 구역에 판매 중인 집의 주소를 찾습니다.
            int state = 0;
            char query1[10000] = { 0, };
            printf(">>> The address of homes for sale in the district Mapo is as follows below: \n");

            snprintf(query1, sizeof(query1), "SELECT address FROM property WHERE address LIKE '%%Mapo-gu%%' AND property_id NOT IN (SELECT property_id FROM sold);");
            state = mysql_query(connection, query1);
            if (state == 0)
            {
                int flag = 1, count = 1;
                sql_result = mysql_store_result(connection);
                printf("\n\n-%s\n", "---Address---");
                while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
                {
                    flag = 0;
                    printf("Address %d: %-100s\n", count++, sql_row[0]);
                }

                if (flag) {
                    printf("No homes for sale in the district Mapo.\n");
                }

                mysql_free_result(sql_result);
                printf("\n");
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }



            while (1) {
                int subtype = 0;
                printf("\n\n---------- Subtypes in TYPE 1 ----------\n");
                printf("\t1. TYPE 1-1.\n");

                printf(">>> Please select a subquery type number. Input 0 to exit: \n");
                scanf("%d", &subtype);

                if (subtype == 0) break;
                else if (subtype == 1) {
                    // TYPE 1-1 쿼리
                    char query2[10000] = { 0, };
                    printf(">>> The address of homes for sale in the district Mapo with price between ₩1,000,000,000 and ₩1,500,000,000 is as follows below: \n");

                    snprintf(query2, sizeof(query2), "SELECT address FROM property WHERE address LIKE '%%Mapo-gu%%' AND property_id NOT IN (SELECT property_id FROM sold) AND price BETWEEN 1000000000 AND 1500000000:");
                    state = mysql_query(connection, query2);
                    if (state == 0)
                    {
                        int flag = 1, count = 1;
                        sql_result = mysql_store_result(connection);
                        printf("\n\n-%s\n", "---Address---");
                        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
                        {
                            flag = 0;
                            printf("Address %d: %-100s\n", count++, sql_row[0]);
                        }

                        if (flag) {
                            printf("No homes for sale in the district Mapo within the specified price range.\n");
                        }

                        mysql_free_result(sql_result);
                        printf("\n");
                    }
                    else {
                        fprintf(stderr, "%s\n", mysql_error(connection));
                    }
                }
```

```
                else {
                    printf(">>> Range Subquery type input is 0 ~ 1. Subypes in TYPE 1 restarts.\n");
                }
            }

        }
        else if (input == 2) {
            // TYPE 2 쿼리
            int state = 0;
            char query1[10000] = { 0, };
            printf(">>> The address of homes for sale in the 8th school district is as follows below: \n");

            snprintf(query1, sizeof(query1), "SELECT address FROM property WHERE school_district = 8 AND property_id NOT IN (SELECT property_id
FROM sold);");
            state = mysql_query(connection, query1);
            if (state == 0)
            {
                int flag = 1, count = 1;
                sql_result = mysql_store_result(connection);
                printf("\n\n-%s\n", "---Address---");
                while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
                {
                    flag = 0;
                    printf("Address %d: %-100s\n", count++, sql_row[0]);
                }

                if (flag) {
                    printf("No homes for sale in the 8th school district.\n");
                }

                mysql_free_result(sql_result);
                printf("\n");
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }

            while (1) {
                int subtype = 0;
                printf("\n\n---------- Subtypes in TYPE 2 ----------\n");
                printf("\t1. TYPE 2-1.\n");

                printf(">>> Please select a subquery type number. Input 0 to exit: \n");
                scanf("%d", &subtype);

                if (subtype == 0) break;
                else if (subtype == 1) {
                    // TYPE 2-1 쿼리
                    char query2[10000] = { 0, };
                    printf(">>> The address of homes for sale in the 8th school district with 4 or more bedrooms and 2 bathrooms is as follows below:
\n");

                    snprintf(query2, sizeof(query2), "SELECT address FROM property WHERE school_district = 8 AND property_id NOT IN (SELECT
property_id FROM sold) AND number_of_bedrooms >= 4 AND number_of_bathrooms = 2;");
                    state = mysql_query(connection, query2);
                    if (state == 0)
                    {
                        int flag = 1, count = 1;
                        sql_result = mysql_store_result(connection);
                        printf("\n\n-%s\n", "---Address---");
                        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
                        {
                            flag = 0;
                            printf("Address %d: %-100s\n", count++, sql_row[0]);
                        }

                        if (flag) {
                            printf("No homes for sale in the 8th school district with 4 or more bedrooms and 2 bathrooms.\n");
                        }

                        mysql_free_result(sql_result);
                        printf("\n");
                    }
```

```
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }
        }
        else {
            printf(">>> Range Subquery type input is 0 ~ 1. Subypes in TYPE 2 restarts.\n");
        }
    }
}

else if (input == 3) {
    // TYPE 3 쿼리
    int state = 0;
    char query[1000] = { 0, };
    printf(">>> The name of the agent who has sold the most properties in the year 2022 by total won value is as follows below: \n");

    snprintf(query, sizeof(query),
        "SELECT a.name "
        "FROM sold s "
        "JOIN agent a ON s.agent_id = a.agent_id "
        "WHERE YEAR(s.sold_date) = 2022 "
        "GROUP BY a.agent_id, a.name "
        "ORDER BY SUM(s.sold_price) DESC "
        "LIMIT 1;");

    state = mysql_query(connection, query);
    if (state == 0) {
        sql_result = mysql_store_result(connection);
        int flag = 1;
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            flag = 0;
            printf("Agent Name: %s\n", sql_row[0]);
        }

        if (flag) {
            printf("No properties sold by any agent in the year 2022.\n");
        }

        mysql_free_result(sql_result);
        printf("\n");
    }
    else {
        fprintf(stderr, "%s\n", mysql_error(connection));
    }

    while (1) {
        int subtype = 0;
        printf("\n\n---------- Subtypes in TYPE 3 ----------\n");
        printf("\t1. TYPE 3-1.\n");
        printf("\t2. TYPE 3-2.\n");

        printf(">>> Please select a subquery type number. Input 0 to exit: \n");
        scanf("%d", &subtype);

        if (subtype == 0) break;
        else if (subtype == 1) {
            // TYPE 3-1 쿼리
            int k;
            printf(">>> Enter the number(k) of top agents to retrieve: \n");
            scanf("%d", &k);
            char query3_1[1000] = { 0, };
            snprintf(query3_1, sizeof(query3_1),
                "SELECT a.agent_id, a.name, SUM(s.sold_price) as total_sold "
                "FROM sold s "
                "JOIN agent a ON s.agent_id = a.agent_id "
                "WHERE YEAR(s.sold_date) = 2023 "
                "GROUP BY a.agent_id, a.name "
                "ORDER BY total_sold DESC "
                "LIMIT %d;", k);

            int state = mysql_query(connection, query3_1);
            if (state == 0) {
                int flag = 1, rank = 1;
```

```
                sql_result = mysql_store_result(connection);
                printf("\n\n--- Top %d agents by total won value in 2023 ---\n", k);
                while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    flag = 0;
                    printf("Rank %d: Agent ID: %s, Name: %s, Total Sold: %s\n", rank++, sql_row[0], sql_row[1], sql_row[2]);
                }

                if (flag) {
                    printf("No agents found for the year 2023.\n");
                }

                mysql_free_result(sql_result);
                printf("\n");
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }

        }
        else if (subtype == 2) {
            // TYPE 3-2 쿼리
            char count_query[1000] = { 0, };
            snprintf(count_query, sizeof(count_query),
                "SELECT FLOOR(COUNT(*) * 0.1) AS limit_value FROM agent;");

            int state = mysql_query(connection, count_query);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                sql_row = mysql_fetch_row(sql_result);
                int limit_value = atoi(sql_row[0]);
                mysql_free_result(sql_result);

                char query3_2[10000] = { 0, };
                snprintf(query3_2, sizeof(query3_2),
                    "SELECT a.agent_id, a.name, COALESCE(SUM(s.sold_price), 0) as total_sold "
                    "FROM agent a "
                    "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2021 "
                    "GROUP BY a.agent_id, a.name "
                    "ORDER BY total_sold ASC "
                    "LIMIT %d;", limit_value);

                state = mysql_query(connection, query3_2);
                if (state == 0) {
                    int flag = 1;
                    sql_result = mysql_store_result(connection);
                    printf("\n\n--- Bottom 10%% agents by total won value in 2021 ---\n");
                    while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                        flag = 0;
                        printf("Agent ID: %s, Name: %s, Total Sold: %s\n", sql_row[0], sql_row[1], sql_row[2]);
                    }

                    if (flag) {
                        printf("No agents found for the year 2021.\n");
                    }

                    mysql_free_result(sql_result);
                    printf("\n");
                }
                else {
                    fprintf(stderr, "%s\n", mysql_error(connection));
                }
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }

        }
    }
}
else if (input == 4) {
    // TYPE 4 쿼리
    int state = 0;
    char query[10000] = { 0, };
```

```
printf(">>> The average selling price and average market time for each agent in 2022 are as follows below: \n");

snprintf(query, sizeof(query),
    "SELECT a.agent_id, a.name, "
    "IFNULL(AVG(s.sold_price), '-') AS avg_selling_price, "
    "IFNULL(AVG(DATEDIFF(s.sold_date, p.listing_date)), '-') AS avg_market_time "
    "FROM agent a "
    "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2022 "
    "LEFT JOIN property p ON s.property_id = p.property_id "
    "GROUP BY a.agent_id, a.name;");

state = mysql_query(connection, query);
if (state == 0) {
    sql_result = mysql_store_result(connection);
    printf("\n\n%-10s %-20s %-20s Won %-20s Days\n", "Agent ID", "Name", "Avg Selling Price", "Avg Market Time");
    int flag = 1;
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
        flag = 0;
        printf("%-10s %-20s %-20s Won %-20s Days\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3]);
    }

    if (flag) {
        printf("No data available for the year 2022.\n");
    }

    mysql_free_result(sql_result);
    printf("\n");
}
else {
    fprintf(stderr, "%s\n", mysql_error(connection));
}

while (1) {
    int subtype = 0;
    printf("\n\n---------- Subtypes in TYPE 4 ----------\n");
    printf("\t1. TYPE 4-1\n");
    printf("\t2. TYPE 4-2\n");

    printf(">>> Please select a subquery type number. Input 0 to exit: \n");
    scanf("%d", &subtype);

    if (subtype == 0) break;
    else if (subtype == 1) {
        // TYPE 4-1 쿼리
        char query4_1[1000] = { 0, };
        snprintf(query4_1, sizeof(query4_1),
            "SELECT a.agent_id, a.name, "
            "IFNULL(MAX(s.sold_price), '-') AS max_selling_price "
            "FROM agent a "
            "LEFT JOIN sold s ON a.agent_id = s.agent_id AND YEAR(s.sold_date) = 2023 "
            "GROUP BY a.agent_id, a.name;");

        int state = mysql_query(connection, query4_1);
        if (state == 0) {
            int flag = 1;
            sql_result = mysql_store_result(connection);
            printf("\n\n--- Maximum selling price of properties sold in 2023 for each agent ---\n");
            while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                flag = 0;
                printf("Agent ID: %s, Name: %s, Max Selling Price: %s Won\n", sql_row[0], sql_row[1], sql_row[2]);
            }

            if (flag) {
                printf("No agents found for the year 2023.\n");
            }

            mysql_free_result(sql_result);
            printf("\n");
        }
        else {
            fprintf(stderr, "%s\n", mysql_error(connection));
        }
```

```
            }
            else if (subtype == 2) {
                // TYPE 4-2 쿼리
                char query4_2[1000] = { 0, };
                snprintf(query4_2, sizeof(query4_2),
                    "SELECT a.agent_id, a.name, "
                    "IFNULL(MAX(DATEDIFF(s.sold_date, p.listing_date)), '-') AS longest_market_time "
                    "FROM agent a "
                    "LEFT JOIN sold s ON a.agent_id = s.agent_id "
                    "LEFT JOIN property p ON s.property_id = p.property_id "
                    "GROUP BY a.agent_id, a.name;");

                int state = mysql_query(connection, query4_2);
                if (state == 0) {
                    int flag = 1;
                    sql_result = mysql_store_result(connection);
                    printf("\n\n--- Longest time the property was on the market for each agent ---\n");
                    while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                        flag = 0;
                        printf("Agent ID: %s, Name: %s, Longest Market Time: %s Days\n", sql_row[0], sql_row[1], sql_row[2]);
                    }

                    if (flag) {
                        printf("No agents found with market time data.\n");
                    }

                    mysql_free_result(sql_result);
                    printf("\n");
                }
                else {
                    fprintf(stderr, "%s\n", mysql_error(connection));
                }
            }
            else {
                printf(">>> Range Subquery type input is 0 ~ 2. Subtypes in TYPE 4 restarts.\n");
            }
        }

    }
    else if (input == 5) {
        // TYPE 5 쿼리
        int state = 0;
        char query[10000] = { 0, };
        char room_types[4][20] = { "studio", "one-bedroom", "multi-bedroom", "detached" };
        char room_descriptions[4][100] = { "most expensive studio", "most expensive one-bedroom", "most expensive multi-bedroom apartment", "most
expensive detached house" };

        for (int i = 0; i < 4; i++) {
            snprintf(query, sizeof(query),
                "SELECT p.photo_type, p.image_url FROM photo p "
                "JOIN property pr ON p.property_id = pr.property_id "
                "WHERE pr.property_type = '%s' "
                "AND pr.price = (SELECT MAX(price) FROM property WHERE property_type = '%s');",
                room_types[i], room_types[i]);
            state = mysql_query(connection, query);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                printf("\n>>> The %s photos are as follows below:\n\n", room_descriptions[i]);
                int flag = 1;
                while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    flag = 0;
                    printf("%s: %s\n", sql_row[0], sql_row[1]);
                }
                if (flag) {
                    printf("No photos found for the %s.\n", room_descriptions[i]);
                }
                mysql_free_result(sql_result);
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }
        }
    }
```

```
        else if (input == 6) {
            int state = 0;
            char property_code[20];
            char buyer_name[50];
            char agent_name[50];
            char seller_name[50];
            char sold_date[20];
            double sold_price;
            int property_id, buyer_id, agent_id, seller_id;

            printf(">>> Record the sale of a property.\n");

            // Get property code
            printf("Enter the property code (property_id): ");
            scanf("%s", property_code);

            // Get sold price
            printf("Enter the sold price: ");
            scanf("%lf", &sold_price);

            // Get buyer name
            printf("Enter the buyer name: ");
            scanf(" %[^\n]", buyer_name);

            // Get agent name
            printf("Enter the agent name: ");
            scanf(" %[^\n]", agent_name);

            // Get seller name
            printf("Enter the seller name: ");
            scanf(" %[^\n]", seller_name);

            // Get sold date
            printf("Enter the sold date (YYYY-MM-DD): ");
            scanf("%s", sold_date);

            // Find property_id
            char query_property[1000];
            snprintf(query_property, sizeof(query_property), "SELECT property_id FROM property WHERE property_id = %s AND property_id NOT IN
(SELECT property_id FROM sold);", property_code);
            state = mysql_query(connection, query_property);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                if ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    property_id = atoi(sql_row[0]);
                }
                else {
                    printf("Invalid property code or the property is already sold.\n");
                    mysql_free_result(sql_result);
                    continue;
                }
                mysql_free_result(sql_result);
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
                return 1;
            }

            // Find buyer_id
            char query_buyer[1000];
            snprintf(query_buyer, sizeof(query_buyer), "SELECT buyer_id FROM buyer WHERE name = '%s';", buyer_name);
            state = mysql_query(connection, query_buyer);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                if ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    buyer_id = atoi(sql_row[0]);
                }
                else {
                    printf("Invalid buyer name.\n");
                    mysql_free_result(sql_result);
                    continue;
                }
                mysql_free_result(sql_result);
```

```
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
                return 1;
            }

            // Find agent_id
            char query_agent[1000];
            snprintf(query_agent, sizeof(query_agent), "SELECT agent_id FROM agent WHERE name = '%s';", agent_name);
            state = mysql_query(connection, query_agent);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                if ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    agent_id = atoi(sql_row[0]);
                }
                else {
                    printf("Invalid agent name.\n");
                    mysql_free_result(sql_result);
                    continue;
                }
                mysql_free_result(sql_result);
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
                return 1;
            }

            // Find seller_id
            char query_seller[1000];
            snprintf(query_seller, sizeof(query_seller), "SELECT seller_id FROM seller WHERE name = '%s';", seller_name);
            state = mysql_query(connection, query_seller);
            if (state == 0) {
                sql_result = mysql_store_result(connection);
                if ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                    seller_id = atoi(sql_row[0]);
                }
                else {
                    printf("Invalid seller name.\n");
                    mysql_free_result(sql_result);
                    continue;
                }
                mysql_free_result(sql_result);
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
                return 1;
            }

            // Insert into sold table
            char query_insert[1000];
            snprintf(query_insert, sizeof(query_insert), "INSERT INTO sold (property_id, agent_id, buyer_id, seller_id, sold_price, sold_date) VALUES (%d,
%d, %d, %d, %lf, '%s');", property_id, agent_id, buyer_id, seller_id, sold_price, sold_date);
            state = mysql_query(connection, query_insert);
            if (state == 0) {
                printf("The sale has been successfully recorded.\n");
            }
            else {
                fprintf(stderr, "%s\n", mysql_error(connection));
            }
        }
        else if (input == 7) {
            int agent_id;
            char name[21]; // Adjusting size to fit VARCHAR(20)
            char contact_info[21]; // Adjusting size to fit VARCHAR(20)
            char query[10000] = { 0 };
            int state = 1;

            printf(">>> Registering a new agent.\n");

            // Prompt user to input a non-duplicate agent_id
            while (1) {
                printf("Agent ID (number between 10000 and 19999): ");
                scanf("%d", &agent_id);
```

```
                // Check if agent_id is within the range
                if (agent_id < 10000 || agent_id > 19999) {
                    printf("Agent ID must be a number between 10000 and 19999. Please re-enter.\n");
                    continue;
                }

                // Check if agent_id is a duplicate
                snprintf(query, sizeof(query), "SELECT COUNT(*) FROM agent WHERE agent_id = %d;", agent_id);
                state = mysql_query(connection, query);
                if (state == 0) {
                    sql_result = mysql_store_result(connection);
                    sql_row = mysql_fetch_row(sql_result);
                    if (atoi(sql_row[0]) == 0) {
                        mysql_free_result(sql_result);
                        break; // If no duplicate found, exit loop
                    }
                    else {
                        printf("The entered Agent ID already exists. Please enter a different ID.\n");
                        mysql_free_result(sql_result);
                    }
                }
                else {
                    fprintf(stderr, "%s\n", mysql_error(connection));
                    return 1;
                }
            }

            printf("Name(lower than 20 characters): ");
            scanf("%s", name);
            printf("Contact Info(lower than 20 characters): ");
            scanf("%s", contact_info);

            snprintf(query, sizeof(query),
                "INSERT INTO agent (agent_id, name, contact_info) VALUES (%d, '%s', '%s');",
                agent_id, name, contact_info);

            state = mysql_query(connection, query);
            if (state == 0) {
                printf("New agent successfully registered.\n");
            }
            else {
                fprintf(stderr, "Error registering new agent: %s\n", mysql_error(connection));
            }
        }
        else {
            printf(">>> Range Subquery type input is 0 ~ 7. Program menu restarts!\n");
        }
    }

    /* 아래의 양식을 참고하여 코드를 작성하였음.
    const char* query = "select * from customer";
    int state = 0;
    state = mysql_query(connection, query);
    if (state == 0) {
        sql_result = mysql_store_result(connection);

        printf("[ SELECT * FROM CUSTOMER ]\n");
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
            printf(" %s  %s  %s  %s\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3]);

        mysql_free_result(sql_result);
    }
    */
    // comment out if you want to persist example db instance.


    while (fgets(line, sizeof(line), fp) != NULL)
        mysql_query(connection, line);                                  // these are DELETEs & DROPs.
    mysql_close(connection);
}
return 0;
}
```