데이터베이스시스템

(CSE4110-02)

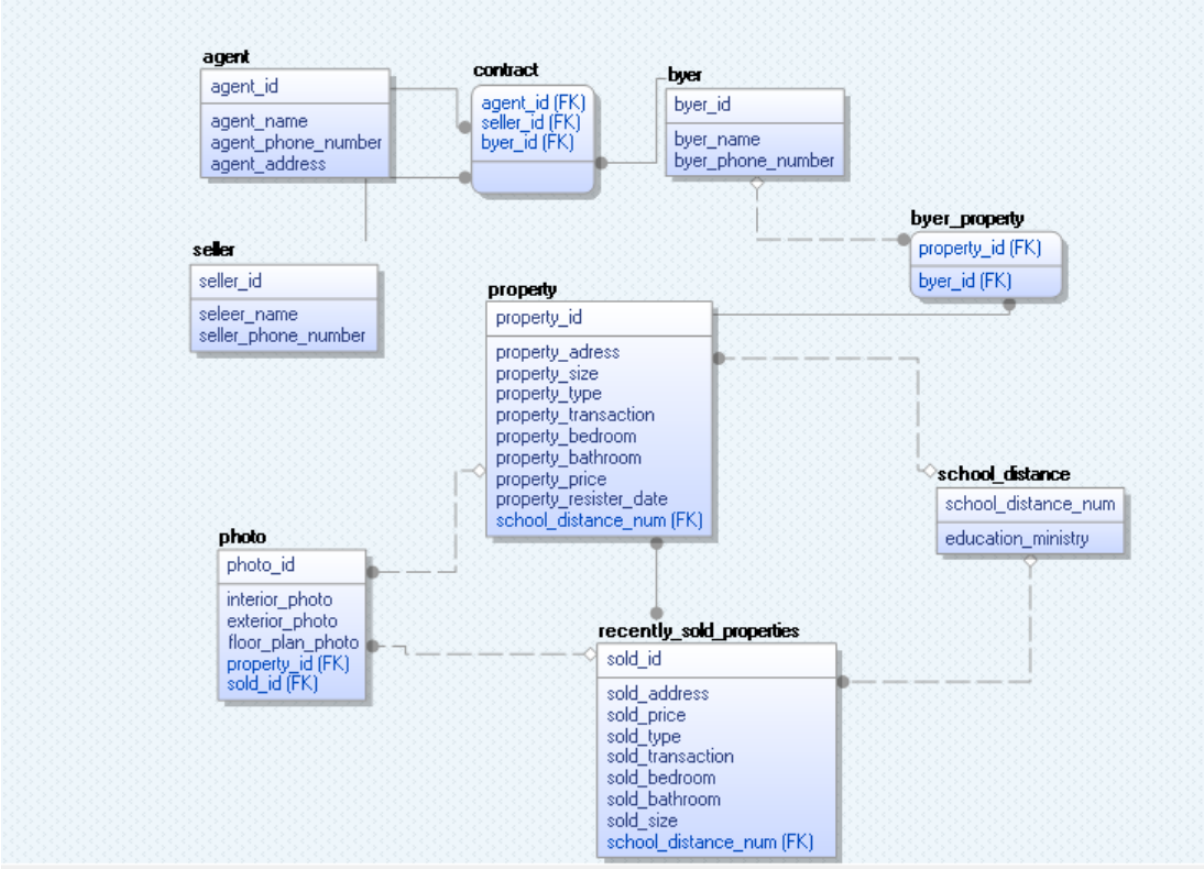
Project2

학번:20211188

이름:방수윤

1. BCNF Decomposition

기존 Schema Diagram



각 logical schema가 BCNF를 만족하는지 확인하고, 만족하지 않는다면 BCNF decomposition을 진행하여 BCNF를 만족하는 schma를 만들겠다.

1) Agent

Agent(agent_id, agent_name, agent_phone_number, agent_address)

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) Agent_id -> agent_name, agent_phone_number, agent_address

Agent_id는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

(2) agent_phone_number -> agent_id, agent_name, agent_address

agent_phone_number 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

(project1에서 중개사는 하나의 전화번호를 가진다고 설정해 두었다.)

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \supseteq \beta$)이거나 또는 α가 superkey 여야 하는데 (1),(2)에서 구한 superkey인 Agent_id와 agent_phone_number 중 하나를 포함하므로 α는 언제나 superkey이다.

따라서 agent는 BCNF를 만족하여 decomposition이 필요없다.

2) property

Property(property_id, property_address, property_size, property_type, property_transaction, property_bedroom, property_bathroom, property_price, property_resister_data, school_distance_num)

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) property_id -> property_address, property_size, property_type, property_transaction, property_bedroom, property_bathroom, property_price, property_resister_dat, school_distance_num)

property_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \supseteq \beta$)이거나 또는 가 superkey 여야하는데 (1)에서 구한 superkey인 property_id를 포함하므로 α는 언제나 superkey이다.

따라서 property는 BCNF를 만족하여 decomposition이 필요없다.

3) Byer

Byer(byer_id, byer_name, byer_phone_number)

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) byer_id -> byer_name, byer_phone_number

byer_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

(2) byer_phone_number -> byer_id, byer_name

byer_phone_number 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

(project1에서 구매자는 하나의 전화번호를 가진다고 설정해 두었다.)

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \rightarrow \beta$)이거나 또는 가 superkey 여야하는데 (1),(2)에서 구한 superkey인 byer_id와 byer_phone_number 중 하나를 포함하므로 α는 언제나 superkey이다.

따라서 byer는 BCNF를 만족하여 decomposition이 필요없다.

4) seller

Seller(seller_id, seller_name, seller_phone_number)

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) seller_id -> seller_name, seller_phone_number

seller_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

(2) seller_phone_number -> seller_name, seller_phone_number

seller_phone_number 는 schema의 모든 attribute을 unique하게 결정하고 있으므로

super key이다.

(project1에서 판매자는 하나의 전화번호를 가진다고 설정해 두었다.)

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \supseteq \beta$)이거나 또는 가 superkey 여야하는데 (1),(2)에서 구한 superkey인 seller_id와 seller_phone_number 중 하나를 포함하므로 α는 언제나 superkey이다.

따라서 seller는 BCNF를 만족하여 decomposition이 필요없다.

5) photo

Photo(photo_id, interior_photo, exterior_photo, floor_plan_photo, property_id, sold_id)

기존 project1에서 구성했던 photo schema는 property_id와 sold_id가 상호 배타적이기 때문에 하나의 사진이 동시에 property와 이미 판매된 매물로 식별할 수 없기에 분리할 필요가 있다. 분리를 한 테이블은 다음과 같다.

Proeprty_photos(photo_id, interior_photo, exterior_photo, floor_plan_photo, property_id)

Sold_property_photos(photo_id, interior_photo, exterior_photo, floor_plan_photo, sold_id)

6) school_distance

School_distance(school_distance_num, education_ministry)

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) school_distance_num -> education_ministry

school_distance_num 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \supseteq \beta$)이거나 또는 가 superkey 여야하는데 (1)에서 구한 superkey인 school_distance_num 를 포함하므로 α는 언제나 superkey이다.

따라서 school_distance는 BCNF를 만족하여 decomposition이 필요없다.

7) recently_sold_properties

Recently_sold_properties(sold_id, sold_address, sold_price, sold_type, sold_transaction, sold_bedroom, sold_bathroom, sold_size, school_distance_num, **sold_date, sold_resister_date**)

**쿼리문을 수행하기 위해 필요한 정보 sold_date, sold_resister_date를 추가하였다.

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) sold_id -> sold_address, sold_price, sold_type, sold_transaction, sold_bedroom, sold_bathroom, sold_size, school_distance_num, sold_date, sold_resister_date

Sold_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

이외의 funtional dependency들은 trivial($\alpha \rightarrow \beta \; or \; \alpha \supseteq \beta$)이거나 또는 가 superkey 여야하는데 (1)에서 구한 superkey인 sold_id 를 포함하므로 α는 언제나 superkey이다.

따라서 recently_sold_properties는 BCNF를 만족하여 decomposition이 필요없다.

8) contract

Contract(contract_id, agent_id, seller_id, byer_id)

**쿼리문을 수행하기 위해 필요한 정보 contract_id를 추가하였다.

BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) contract_id -> agent_id, seller_id, byer_id

contract_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

이외의 funtional dependency들은 trivial($\alpha \supseteq \beta \; or \; \alpha \rightarrow \beta$)이거나 또는 가 superkey 여야하는데 (1)에서 구한 superkey인 contract_id 를 포함하므로 α는 언제나 superkey이다.

따라서 contract는 BCNF를 만족하여 decomposition이 필요없다.

9) byer_property

Byer_propterty(property_id, byer_id)

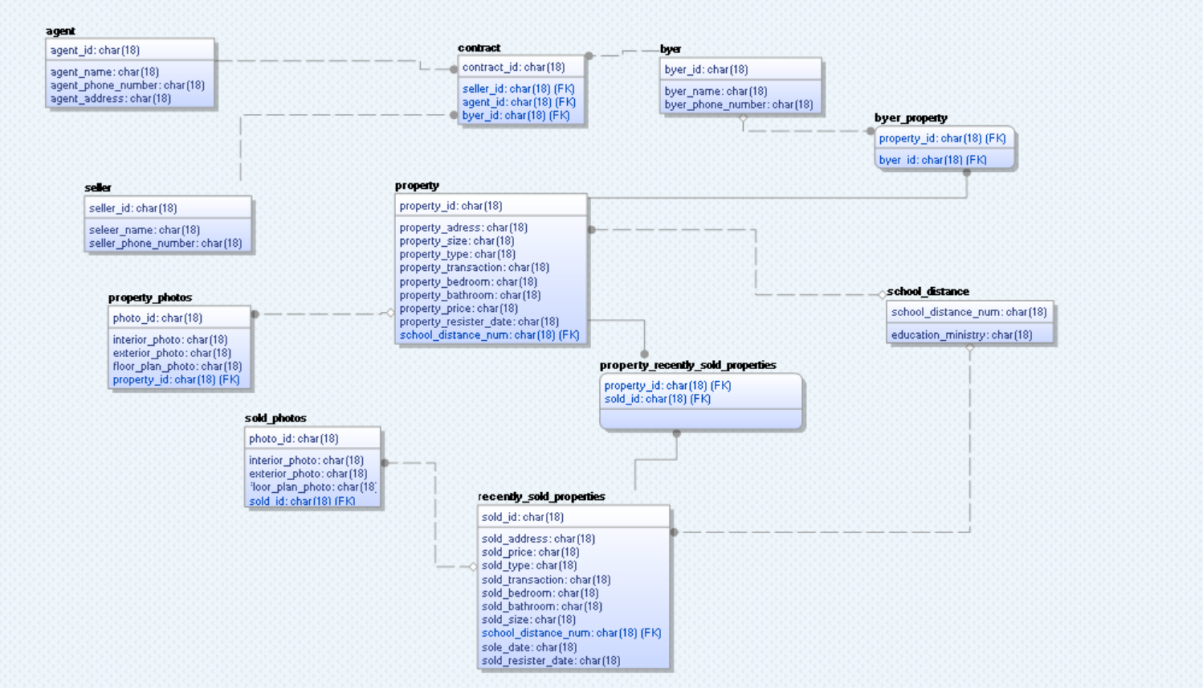
BCNF의 조건을 만족하는지 확인 절차를 거친다.

(1) property_id -> byer_id

property_id 는 schema의 모든 attribute을 unique하게 결정하고 있으므로 super key이다.

따라서 byer는 BCNF를 만족하여 decomposition이 필요없다.

BCNF Decomposition이 필요한 항목이 없고, 최종적인 logical schema diagram은 다음과 같다.



2. physical shcema diagram

각 schema에 대한 설명은 다음과 같다. 동일한 schema에 대해서는 필요한 내용을 추가하여 project1에서의 설명을 차용했다.

(1) Property

중개사에서 관리하는 부동산 매물의 정보를 저장하고 있는 set이다.

-property_id: 매물별로 부여 받은 고유의 id를 의미한다.

-property_address: 매물의 위치를 나타내는 주소를 의미한다. 시, 도와 함께 상세주소에 관한 value를 갖는다.

-property_size: 매물의 크기를 나타낸다. 평수를 value로 갖는다.

-property_type: 매물의 종류를 나타낸다. 오피스텔, 빌라, 아파트 등의 종류 중 어떤 type인지를 저장한다.

-property_transaction: 매물의 전세/매매/월세와 같은 부동산 거래 유형을 value로 갖는다.

-property_bedroom: 매물의 침실 개수를 갖는다.

-property_bathroom: 매물의 화장실 개수를 가진다.

-property_price: 매물의 가격을 value로 갖는다.

-property_resister_date: 매물 등록 년도, 월, 일, 시간의 형태식의 value를 갖는다.

-school_distance_num: 서울에서 1~11학군 중 해당 매물과 가장 가까운 학군 number을 의미한다.

property에서의 primary key는 property_id이다.

(2) agent

부동산 매물을 거래하는 중개사에 관한 정보를 담고 있는 set이다.

-agent_id: 중개사의 가입 아이디로 고유번호를 나타낸다.

-agent_name: 중개사의 이름을 나타낸다.

-agent_phone_number: 중개사의 전화번호를 나타낸다. 중개사는 하나의 전화번호를 가진다.

-agent_address: 중개소의 위치를 나타낸다.

agent의 primary key는 agent_id이다.

(3) byer

부동산 매물을 구매하는 구매자에 관한 정보를 담고 있는 set이다.

-byer_id: 구매자의 고유 id를 나타낸다.

-byer_name: 구매자의 이름을 의미한다.

-byer_phone_number: 구매자의 전화번호를 의미한다. 구매자는 하나의 전화번호만을 가진다.

byer의 primary key는 byer_id이다.

(4) seller

부동산 매물을 판매하는 판매자에 관한 정보를 저장하는 set이다.

-seller_id: 판매자의 고유한 id를 나타낸다.

-seller_name: 판매자의 이름을 나타낸다.

-seller_phone_number: 판매자의 전화번호를 의미한다. 판매자는 하나의 전화번호만을 가진다.

seller의 primary key는 seller_id이다.

(5) property_photos

매물의 사진 정보를 저장하는 set이다.

-photo_id: 매물 사진의 고유한 id를 나타낸다.

-interior_photo: 매물의 외부사진을 나타낸다.

-exterior_photo: 매물의 내부사진을 나타낸다.

-floor_plan_photo: 매물의 바닥 도면 사진을 나타낸다.

-property_id: 매물별로 부여 받은 고유의 id를 의미한다.

Property_photos의 primary key는 photo_id이다.

(6) sold_photos

판매된 부동산의 사진 정보를 저장하는 set이다.

-photo_id: 매물 사진의 고유한 id를 나타낸다.

-interior_photo: 매물의 외부사진을 나타낸다.

-exterior_photo: 매물의 내부사진을 나타낸다.

-floor_plan_photo: 매물의 바닥 도면 사진을 나타낸다.

-sold_id: 최근 판매된 매물의 고유한 id이다.

Sold_photos의 primary key는 photo_id이다.

(7) school_distance

매물과 가까운 학교 정보를 저장하는 set이다.

-school_distance_num: 서울에서 1~11학군 중 해당 매물과 가장 가까운 학군 number을 의미한다.

-education_ministry: 해당 학군의 교육청 이름을 의미한다.

school_distance의 primary key는 school_distance_num이다.

(8) recently sold properties

같은 중개인이 최근에 판매했던 매물의 정보를 저장하는 set이다.

-sold_id: 최근 판매된 매물의 고유한 id이다.

-sold_address: 최근 판매된 매물의 주소를 나타낸다.

-sold_price: 최근 판매된 매물의 가격을 나타낸다.

-sold_type: 최근 판매된 매물의 종류를 나타낸다. 오피스텔, 빌라, 아파트 등의 종류 중 어떤 type 인지를 저장한다.

-sold_transaction: 최근 판매된 매물의 전세/매매/월세와 같은 부동산 거래 유형을 value로 갖는다.

-sold__bedroom: 최근 판매된 매물의 침실 개수를 갖는다.

-sold_bathroom: 최근 판매된 매물의 화장실 개수를 갖는다.

-sold_size: 최근 판매된 매물의 평수를 나타낸다.

-sole_date: 최근 판매된 매물의 거래 날(년도, 월, 일)을 담고있다.

-school_distance_num: 서울에서 1~11학군 중 해당 매물과 가장 가까운 학군 number을 의미한다.

-sold_resister_date: 최근 판매된 매물이 첫 개시된 날(년도, 월, 일)을 담고 있다.

recently sold properties의 primary key는 sold_id이다.

(9) contract

구매자, 중개사, 판매자의 계약 관계를 저장하고 있는 schema다.

-Agent_id: 중개사의 가입 아이디로 고유번호를 나타낸다.

-seller_id: 판매자의 고유한 id를 나타낸다.

-byer_id: 구매자의 고유 id를 나타낸다.

-contract_id: 거래의 고유 id를 나타낸다.

Contract의 primary key는 contract_id이다.

(10) byer_property

구매자와 매물 정보 간의 관계를 저장하고 있는 schema다.

-byer_id: 구매자의 고유 id를 나타낸다.

-property_id: 매물별로 부여 받은 고유의 id를 의미한다.

Byer_property의 primary key는 property_id이다.

3. Queries

(type1)

마포에 있는 매물 주소를 찾는 상황이다.

```
SELECT property_address
FROM property
WHERE property_address LIKE '%Mapo-gu%';
```

마포에 있는 property의 주소를 구하는 query이다.

Type 1을 구현한 mysql 코드는 위와 같다.

Type 1 에서는 따로 입력을 받지 않는다.

(type1-1)

```
SELECT property_address
FROM property
WHERE property_address LIKE '%Mapo-gu%'
  AND property_price BETWEEN 1000000000 AND 1500000000;
```

마포에 있는 property 중 가격이 1,000,000,000~1,500,000,000 범위에 있는 주소를 구하는 query이다.

Type 1-1을 구현한 mysql 코드는 위와 같다.

Type 1-1 에서는 따로 입력을 받지 않는다.

(type2)

8학군에 있는 매물의 주소를 찾는 상황이다.

```
SELECT property_address
FROM property
JOIN school_distance USING(school_distance_num)
WHERE school_distance_num = 8;
```

8학군인 property의 주소를 구하는 query이다.

Type 2을 구현한 mysql 코드는 위와 같다.

Type 2 에서는 따로 입력을 받지 않는다.

(type2-1)

```
SELECT property_address
FROM property
JOIN school_distance USING(school_distance_num)
WHERE school_distance_num = 8
  AND property_bedroom >= 4
  AND property_bathroom = 2;
```

8학군인 property 중 침실이 4개 이상이고 화장실이 2개인 매물의 주소를 구하는 query이다.

Type 2-1을 구현한 mysql 코드는 위와 같다.

Type 2-1 에서는 따로 입력을 받지 않는다.

(type3)

2022년에 매물을 팔아서 가장 많은 돈을 번 중개사를 찾는 상황이다.

```
SELECT a.agent_id, a.agent_name, SUM(rsp.sold_price) AS total_value_2022
FROM contact
NATURAL JOIN recently_sold_properties rsp
JOIN agent a USING(agent_id)
WHERE JSON_EXTRACT(rsp.sold_date, '$.year') = '2022'
GROUP BY a.agent_id, a.agent_name
ORDER BY total_value_2022 DESC
LIMIT 1;
```

2022년에 매물을 팔아서 가장 돈을 많이 번 중개사를 구하는 query이다.

Type 3을 구현한 mysql 코드는 위와 같다.

Type 3 에서는 따로 입력을 받지 않는다.

(type3-1)

```sql
WITH RankedAgents AS (
    SELECT
        a.agent_id,
        a.agent_name,
        SUM(rsp.sold_price) AS total_value_2023,
        DENSE_RANK() OVER (ORDER BY SUM(rsp.sold_price) DESC) AS rank_value
    FROM
        contact c
        JOIN agent a ON c.agent_id = a.agent_id
        JOIN recently_sold_properties rsp ON c.seller_id = rsp.sold_id
    WHERE
        YEAR(rsp.sold_date) = 2023
    GROUP BY
        a.agent_id, a.agent_name
)
SELECT
    agent_id,
    agent_name,
    total_value_2023
FROM
    RankedAgents
WHERE
    rank_value <= k;
```

2023년에 매물을 판 중개사 중 상위 랭크 k안에 드는 agent들을 구하는 query이다.

Type 3-1을 구현한 mysql 코드는 위와 같다.

실제로도 k(rank)에 대응하는 숫자가 입력되어야 작동한다. 추후에 c++ code를 구현 할 때 k라는 숫자를 직접 입력 받아 mysql로 넘겨주었다.

(type3-2)

```sql
WITH AgentSales2021 AS (
    SELECT a.agent_id, a.agent_name, SUM(rsp.sold_price) AS total_value_2021
    FROM contact
    NATURAL JOIN recently_sold_properties rsp
    JOIN agent a USING(agent_id)
    WHERE JSON_UNQUOTE(JSON_EXTRACT(rsp.sold_date, '$.year')) = '2021'
    GROUP BY a.agent_id, a.agent_name
),
AgentSalesCount AS (
    SELECT COUNT(*) AS total_agents FROM AgentSales2021
),
Bottom10Percent AS (
    SELECT *, NTILE(10) OVER (ORDER BY total_value_2021 ASC) AS decile
    FROM AgentSales2021
)
SELECT agent_id, agent_name, total_value_2021
FROM Bottom10Percent
WHERE decile = 1;
```

2021년에 매물을 판 중개사들 중 하위 10%를 구하는 query이다.

Type 3-2을 구현한 mysql 코드는 위와 같다.

Type 3-2 에서는 따로 입력을 받지 않는다.

(type4)

2022년에 판매된 부동산(property)의 평균 판매 가격과 부동산이 시장에 나와 있던 평균 기간을 계산하는 상황이다.

```sql
SELECT a.agent_id, a.agent_name,
       AVG(rsp.sold_price) AS avg_selling_price_2022,
       AVG(DATEDIFF(rsp.sold_date, rsp.sold_resister_date)) AS avg_time_on_market
FROM recently_sold_properties rsp
JOIN contact c ON rsp.sold_id = c.contact_id
JOIN agent a ON c.agent_id = a.agent_id
WHERE YEAR(rsp.sold_date) = 2022
GROUP BY a.agent_id, a.agent_name
LIMIT 0, 1000;
```

2022년에 판매된 부동산(property)의 평균 판매 가격과 부동산이 시장에 나와 있던 평균 기간을 계산하는 query이다.

Type 4을 구현한 mysql 코드는 위와 같다.

Type 4 에서는 따로 입력을 받지 않는다.

(type4-1)

```
SELECT a.agent_id, a.agent_name, MAX(rsp.sold_price) AS max_selling_price_2023
FROM recently_sold_properties rsp
JOIN contact c ON rsp.sold_id = c.contact_id
JOIN agent a ON c.agent_id = a.agent_id
WHERE YEAR(rsp.sold_date) = 2023
GROUP BY a.agent_id, a.agent_name;
```

agent 별로 2023년에 판매된 부동산(property)의 최고가를 비교하는 query이다.

Type 4-1을 구현한 mysql 코드는 위와 같다.

Type 4-1 에서는 따로 입력을 받지 않는다.

(type4-2)

```
SELECT a.agent_id, a.agent_name,
       MAX(DATEDIFF(rsp.sold_date, rsp.sold_resister_date)) AS longest_time_on_market
FROM recently_sold_properties rsp
JOIN contact c ON rsp.sold_id = c.contact_id
JOIN agent a ON c.agent_id = a.agent_id
GROUP BY a.agent_id, a.agent_name;
```

agent 별로 매물로 가장 오래 올라와 있던 매물을 비교하는 query이다.

Type 4-2을 구현한 mysql 코드는 위와 같다.

Type 4-2 에서는 따로 입력을 받지 않는다.

(type5)

```sql
SELECT property_type, interior_photo, exterior_photo, floor_plan_photo
FROM (
    SELECT 'Studio' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price
    FROM property_photos pp
    JOIN property p ON pp.property_id = p.property_id
    WHERE p.property_type = 'Studio'
    ORDER BY p.property_price DESC
    LIMIT 1
) AS studio
UNION ALL
SELECT property_type, interior_photo, exterior_photo, floor_plan_photo
FROM (
    SELECT 'One-bedroom' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price
    FROM property_photos pp
    JOIN property p ON pp.property_id = p.property_id
    WHERE p.property_type = 'Apartment' AND p.property_bedroom = 1
    ORDER BY p.property_price DESC
    LIMIT 1
) AS one_bedroom
UNION ALL
SELECT property_type, interior_photo, exterior_photo, floor_plan_photo
FROM (
    SELECT 'Multi-bedroom' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price
    FROM property_photos pp
    JOIN property p ON pp.property_id = p.property_id
    WHERE p.property_type = 'Apartment' AND p.property_bedroom > 1
    ORDER BY p.property_price DESC
    LIMIT 1
) AS multi_bedroom
UNION ALL
SELECT property_type, interior_photo, exterior_photo, floor_plan_photo
FROM (
    SELECT 'Detached House' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price
    FROM property_photos pp
    JOIN property p ON pp.property_id = p.property_id
    WHERE p.property_type = 'House'
    ORDER BY p.property_price DESC
    LIMIT 1
) AS detached_house
LIMIT 4;
```

가장 비싼 스튜디오, 원룸, 다세대 아파트, 독립 주택의 사진을 각각 데이터베이스에서 보여주는 query이다.

Type 5 을 구현한 mysql 코드는 위와 같다.

Type 5 에서는 따로 입력을 받지 않는다.

(type6)

```sql
START TRANSACTION;
INSERT INTO recently_sold_properties (sold_address, sold_price, sold_type, sold_transaction, sold_bedroom, sold_bathroom, sold_size, school_distance_num, sold_date, sold_resister_date)
SELECT property_address, 1000000000, property_type, 'Sale', property_bedroom, property_bathroom, property_size, school_distance_num, '2023-06-01', property_resister_date
FROM property WHERE property_id = 1;
INSERT INTO byer_property (property_id, byer_id)
SELECT 1, 2 FROM DUAL
WHERE NOT EXISTS (SELECT 1 FROM byer_property WHERE property_id = 1 AND byer_id = 2);
INSERT INTO contact (agent_id, seller_id, byer_id)
VALUES (5, 4, 2);

COMMIT;
```

판매 중으로 등록된 부동산의 판매 기록 저장하는 query이다.

이 쿼리를 실행시키기 위해 추가적으로 필요한 정보인 sold_price, byer_id, seller_id, agent_id, sold_date만 예시를 위해 넣어 놓은 상태이다.

실제로는 sold_price, byer_id, seller_id, agent_id, sold_date 에 대응하는 정보가 직접 입력

되어야 작동한다.

추후에 c++ code를 구현 할 때 직접 입력 받아 mysql로 넘겨주었다.

(type7)

```
INSERT INTO agent (agent_name, agent_phone_number, agent_address)
VALUES ('New Agent Name', '010-1234-5678', '1234 New Address, Seoul');
```

데이터베이스에 새로운 에이전트 추가하는 상황이다.

새로운 agent에서 추가에 필요한 정보인 agent_name, agent_phone_number, agent_address를 예시를 위해 넣어 놓은 상태이다.

실제로는 agent_name, agent_phone_number, agent_address 에 대응하는 정보가 직접 입력되어야 작동한다.

추후에 c++ code를 구현 할 때 직접 입력 받아 mysql로 넘겨주었다.

4. Code Implemetation

(type1)

마포에 있는 매물 주소를 찾는 상황이다.

```
if (type == 1) {
    printf("---- TYPE 1 ----₩n");
    printf("** Find address of homes for sale in the distinct mapo. **₩n");
    query = "SELECT property_address ";
    query += "FROM property ";
    query += "WHERE property_address LIKE '%mapo-gu%'";
    const char* cquery = query.c_str();
    state = mysql_query(connection, cquery);
    if (state != 0) {
        printf("Query Error: %s₩n", mysql_error(connection));
    }
    else {
        sql_result = mysql_store_result(connection);
        if (sql_result) {
            while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                printf("property_address : %s₩n", sql_row[0]);
            }
            mysql_free_result(sql_result);
        }
        else {
            printf("No results found.₩n");
        }
    }
    printf("₩n");
    printf("----- Subtypes in TYPE 1 -----₩n");
    printf("   1.  TYPE 1-1.₩n");
    int sub_num; cin >> sub_num;
```

마포에 있는 property의 주소를 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type1-1)

```
if (sub_num == 1) {
    printf("---- TYPE 1-1 ----₩n");
    printf("** Find the costing between ₩ 1,000,000,000 and ₩1,500,000,000. **₩n");
    query = "SELECT property_address ";
    query += "FROM property ";
    query += "WHERE property_address LIKE '%mapo-gu%' ";
    query += "AND property_price BETWEEN 1000000000 AND 1500000000";
    cquery = query.c_str();
    state = mysql_query(connection, cquery);
    if (state != 0) {
        printf("Query Error: %s₩n", mysql_error(connection));
    }
    else {
        sql_result = mysql_store_result(connection);
        if (sql_result) {
            while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
                printf("property_address : %s₩n", sql_row[0]);
            }
            mysql_free_result(sql_result);
        }
        else {
            printf("No results found.₩n");
        }
    }
    printf("₩n");
}
```

마포에 있는 property 중 가격이 1,000,000,000~1,500,000,000 범위에 있는 주소를 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type2)

8학군에 있는 매물의 주소를 찾는 상황이다.

```
printf("---- TYPE 2 ----₩n");
printf("** Find the address of homes for sale in the 8th school district. **₩n");
query = "SELECT property_address ";
query += "FROM property ";
query += "JOIN school_distance USING(school_distance_num) ";
query += "WHERE school_distance_num=8";
const char* cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("%s₩n", sql_row[0]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

8학군인 property의 주소를 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type2-1)

```
printf("---- TYPE 2-1 ----₩n");
printf("** Find properties with 4 or more bedrooms and 2 bathrooms **₩n");
query = "SELECT property_address ";
query += "FROM property ";
query += "JOIN school_distance USING(school_distance_num) ";
query += "WHERE school_distance_num=8 ";
query += "AND property_bedroom >= 4 ";
query += "AND property_bathroom=2";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("property_address : %s₩n", sql_row[0]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

8학군인 property 중 침실이 4개 이상이고 화장실이 2개인 매물의 주소를 구하는 query 이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type3)

2022년에 매물을 팔아서 가장 많은 돈을 번 중개사를 찾는 상황이다.

```
printf("---- TYPE III ----₩n");
printf("** Find the name of the agent who has sold the most properties in the year 2022 by total won value **₩n");
query = "SELECT a.agent_id, a.agent_name, SUM(rsp.sold_price) AS total_value_2022 ";
query += "FROM contact ";
query += "NATURAL JOIN recently_sold_properties rsp ";
query += "JOIN agent a USING(agent_id) ";
query += "WHERE YEAR(rsp.sold_date) = 2022 ";
query += "GROUP BY a.agent_id, a.agent_name ";
query += "ORDER BY total_value_2022 DESC ";
query += "LIMIT 1";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩ntotal_value_2022 : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

2022년에 매물을 팔아서 가장 돈을 많이 번 중개사를 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type3-1)

```
printf("--- TYPE 3-1 ---₩n");
printf("** Find the top k agents in the year 2023 by total won value. **₩n");
printf("Which k? :");
int x; cin >> x;
query = "WITH RankedAgents AS (";
query += "SELECT a.agent_id, a.agent_name, SUM(rsp.sold_price) AS total_value_2023, ";
query += "DENSE_RANK() OVER (ORDER BY SUM(rsp.sold_price) DESC) AS rank_value ";
query += "FROM contact c JOIN agent a ON c.agent_id = a.agent_id ";
query += "JOIN recently_sold_properties rsp ON c.seller_id = rsp.sold_id ";
query += "WHERE YEAR(rsp.sold_date) = 2023 ";
query += "GROUP BY a.agent_id, a.agent_name ";
query += ") SELECT agent_id, agent_name, total_value_2023 ";
query += "FROM RankedAgents ";
query += "WHERE rank_value < " + to_string(x);
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩ntotal_value_2023 : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

2023년에 매물을 판 중개사 중 상위 랭크 k안에 드는 agent들을 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

랭크 K를 입력받고 이를 바탕으로 쿼리문을 수행한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type3-2)

```
printf("--- TYPE 3-2 ---₩n");
printf("** Find the bottom 10percent agents in the year 2021 by total won value. **₩n");
query = "WITH AgentSales2021 AS (";
query += "SELECT a.agent_id, a.agent_name, SUM(rsp.sold_price) AS total_value_2021 ";
query += "FROM contact ";
query += "NATURAL JOIN recently_sold_properties rsp ";
query += "JOIN agent a USING(agent_id) ";
query += "WHERE YEAR(rsp.sold_date) = 2021 ";
query += "GROUP BY a.agent_id, a.agent_name ";
query += "), ";
query += "AgentSalesCount AS (";
query += "SELECT COUNT(*) AS total_agents FROM AgentSales2021 ";
query += "), ";
query += "Bottom10Percent AS (";
query += "SELECT *, NTILE(10) OVER (ORDER BY total_value_2021 ASC) AS decile ";
query += "FROM AgentSales2021 ";
query += ") ";
query += "SELECT agent_id, agent_name, total_value_2021 ";
query += "FROM Bottom10Percent ";
query += "WHERE decile = 1;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩ntotal_value_2021 : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

2021년에 매물을 판 중개사들 중 하위 10%를 구하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type4)

2022년에 판매된 부동산(property)의 평균 판매 가격과 부동산이 시장에 나와 있던 평균 기간을 계산하는 상황이다.

```
printf("---- TYPE IV ----₩n");
printf("** Compute the average selling price of 2022, average time the property was on the market. **₩n");
query = "SELECT a.agent_id, a.agent_name, ";
query += "AVG(rsp.sold_price) AS avg_selling_price_2022, ";
query += "AVG(DATEDIFF(rsp.sold_date, rsp.sold_resister_date)) AS avg_time_on_market ";
query += "FROM recently_sold_properties rsp ";
query += "JOIN contact c ON rsp.sold_id = c.contact_id ";
query += "JOIN agent a ON c.agent_id = a.agent_id ";
query += "WHERE YEAR(rsp.sold_date) = 2022 ";
query += "GROUP BY a.agent_id, a.agent_name ";
query += "LIMIT 0, 1000;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩navg_selling_price_2022 : %s₩navg_time_on_market : %s₩n",
                sql_row[0], sql_row[1], sql_row[2], sql_row[3]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

2022년에 판매된 부동산(property)의 평균 판매 가격과 부동산이 시장에 나와 있던 평균 기간을 계산하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type4-1)

```
printf("---- TYPE IV-1 ----₩n");
printf("** Compute max selling price of properties sold in 2023 for each agent **₩n");
query = "SELECT a.agent_id, a.agent_name, ";
query += "MAX(rsp.sold_price) AS max_selling_price_2023 ";
query += "FROM recently_sold_properties rsp ";
query += "JOIN contact c ON rsp.sold_id = c.contact_id ";
query += "JOIN agent a ON c.agent_id = a.agent_id ";
query += "WHERE YEAR(rsp.sold_date) = 2023 ";
query += "GROUP BY a.agent_id, a.agent_name;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩nmax_selling_price_2023 : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

agent 별로 2023년에 판매된 부동산(property)의 최고가를 비교하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type4-2)

```
printf("---- TYPE IV-2 ----₩n");
printf("** Compute the longest time the property was on the market for each agent **₩n");
query = "SELECT a.agent_id, a.agent_name, ";
query += "MAX(DATEDIFF(rsp.sold_date, rsp.sold_resister_date)) AS longest_time_on_market ";
query += "FROM recently_sold_properties rsp ";
query += "JOIN contact c ON rsp.sold_id = c.contact_id ";
query += "JOIN agent a ON c.agent_id = a.agent_id ";
query += "GROUP BY a.agent_id, a.agent_name;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("agent_id : %s₩nagent_name : %s₩nlongest_time_on_market : %s₩n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

agent 별로 매물로 가장 오래 올라와 있던 매물을 비교하는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type5)

```
printf("---- TYPE V ----₩n");
printf("** Show photos of the most expensive studio, one-bedroom, multi-bedroom apartment(s), and detached house(s), respectively. **₩n");
query = "SELECT property_type, interior_photo, exterior_photo, floor_plan_photo, property_price ";
query += "FROM ( ";
query += "SELECT 'Studio' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price ";
query += "FROM property_photos pp JOIN property p ON pp.property_id = p.property_id ";
query += "WHERE p.property_type = 'Studio' ORDER BY p.property_price DESC LIMIT 1 ";
query += ") AS studio ";
query += "UNION ALL ";
query += "SELECT property_type, interior_photo, exterior_photo, floor_plan_photo, property_price ";
query += "FROM ( ";
query += "SELECT 'One-bedroom' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price ";
query += "FROM property_photos pp JOIN property p ON pp.property_id = p.property_id ";
query += "WHERE p.property_type = 'Apartment' AND p.property_bedroom = 1 ORDER BY p.property_price DESC LIMIT 1 ";
query += ") AS one_bedroom ";
query += "UNION ALL ";
query += "SELECT property_type, interior_photo, exterior_photo, floor_plan_photo, property_price ";
query += "FROM ( ";
query += "SELECT 'Multi-bedroom' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price ";
query += "FROM property_photos pp JOIN property p ON pp.property_id = p.property_id ";
query += "WHERE p.property_type = 'Apartment' AND p.property_bedroom > 1 ORDER BY p.property_price DESC LIMIT 1 ";
query += ") AS multi_bedroom ";
query += "UNION ALL ";
query += "SELECT property_type, interior_photo, exterior_photo, floor_plan_photo, property_price ";
query += "FROM ( ";
query += "SELECT 'Detached House' AS property_type, pp.interior_photo, pp.exterior_photo, pp.floor_plan_photo, p.property_price ";
query += "FROM property_photos pp JOIN property p ON pp.property_id = p.property_id ";
query += "WHERE p.property_type = 'House' ORDER BY p.property_price DESC LIMIT 1 ";
query += ") AS detached_house ";
query += "LIMIT 4;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Query Error: %s₩n", mysql_error(connection));
}
else {
    sql_result = mysql_store_result(connection);
    if (sql_result) {
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
            printf("property_type : %s₩ninterior_photo : %s₩nexterior_photo : %s₩nfloor_plan_photo : %s₩nproperty_price : %s₩n",
                sql_row[0], sql_row[1], sql_row[2], sql_row[3], sql_row[4]);
        }
        mysql_free_result(sql_result);
    }
    else {
        printf("No results found.₩n");
    }
}
```

가장 비싼 스튜디오, 원룸, 다세대 아파트, 독립 주택의 사진을 각각 데이터베이스에서 보여주는 query이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결하고, 연결이 성공적으로 되었다면, 결과로 나온 table의 tuple 값을 읽어 들여 console 창에 출력한다. 마지막으로, mysql_free_result 함수를 사용하여 결과 table을 free해준다.

(type6)

```
printf("---- TYPE 6 ----₩n");
printf("** Record the sale of a property that had been listed as being available. entails storing the sales price, the buyer, the seller, the agent (if any), and the date. **₩n");

int property_id, buyer_id, seller_id, agent_id;
double sold_price;
string sold_date;

cout << "Enter property ID: ";
cin >> property_id;
cout << "Enter sold price: ";
cin >> sold_price;
cout << "Enter buyer ID: ";
cin >> buyer_id;
cout << "Enter seller ID: ";
cin >> seller_id;
cout << "Enter agent ID (if any, else 0): ";
cin >> agent_id;
cout << "Enter sold date (YYYY-MM-DD): ";
cin >> sold_date;
string query = "START TRANSACTION;";
const char* cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Transaction Start Error: %s₩n", mysql_error(connection));
    return 1;
}
query = "INSERT INTO recently_sold_properties (sold_address, sold_price, sold_type, sold_transaction, sold_bedroom, sold_bathroom, sold_size, school_distance_num, sold_date, sold_resister_date) ";
query += "SELECT property_address, " + to_string(sold_price) + ", property_type, 'Sale', property_bedroom, property_bathroom, property_size, school_distance_num, '" + sold_date + "', property_resister_date ";
query += "FROM property WHERE property_id = " + to_string(property_id) + ";";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Insert Error (recently_sold_properties): %s₩n", mysql_error(connection));
    mysql_query(connection, "ROLLBACK;");
    return 1;
}

query = "SELECT COUNT(*) FROM byer_property WHERE property_id = " + to_string(property_id) + " AND byer_id = " + to_string(buyer_id) + ";";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Select Error (byer_property): %s₩n", mysql_error(connection));
    mysql_query(connection, "ROLLBACK;");
    return 1;
}
sql_result = mysql_store_result(connection);
if (!sql_result) {
    printf("Store Result Error (byer_property): %s₩n", mysql_error(connection));
    mysql_query(connection, "ROLLBACK;");
    return 1;
}
sql_row = mysql_fetch_row(sql_result);
int count = atoi(sql_row[0]);
mysql_free_result(sql_result);

if (count == 0) {
    query = "INSERT INTO byer_property (property_id, byer_id) VALUES (" + to_string(property_id) + ", " + to_string(buyer_id) + ");";
    cquery = query.c_str();
    state = mysql_query(connection, cquery);
    if (state != 0) {
        printf("Insert Error (byer_property): %s₩n", mysql_error(connection));
        mysql_query(connection, "ROLLBACK;");
        return 1;
    }
}
else {
    printf("Entry already exists in byer_property, skipping insert.₩n");
}

query = "INSERT INTO contact (agent_id, seller_id, byer_id) VALUES (";
query += to_string(agent_id) + ", " + to_string(seller_id) + ", " + to_string(buyer_id) + ");";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Insert Error (contact): %s₩n", mysql_error(connection));
    mysql_query(connection, "ROLLBACK;");
    return 1;
}
query = "COMMIT;";
cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Commit Error: %s₩n", mysql_error(connection));
    mysql_query(connection, "ROLLBACK;");
    return 1;
}

printf("Transaction completed successfully.₩n");
```

판매 중으로 등록된 부동산의 판매 기록 저장하는 query이다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

그리고, mysql_query 함수를 통해 MySQL과 연결한다.

Property에 있는 정보에 추가적 정보인 sold_price, byer_id, seller_id, agent_id, sold_date

를 받아서 contract, recently_sold_properties에 판매기록을 저장하게 된다.

만약 정상적으로 작동되었다면 "Transaction completed successfully"라는 문구가 뜬다.

(type7)

```
printf("---- TYPE VII ----₩n");
printf("** Add a new agent to the database **₩n");
string agent_name;
string agent_phone_number;
string agent_address;
cout << "Enter agent name: ";
cin.ignore();
getline(cin, agent_name);
cout << "Enter agent phone number: ";
getline(cin, agent_phone_number);
cout << "Enter agent address: ";
getline(cin, agent_address);

query = "INSERT INTO agent (agent_name, agent_phone_number, agent_address) ";
query += "VALUES ('" + agent_name + "', '" + agent_phone_number + "', '" + agent_address + "');";

const char* cquery = query.c_str();
state = mysql_query(connection, cquery);
if (state != 0) {
    printf("Insert Error (agent): %s₩n", mysql_error(connection));
}
else {
    printf("New agent added successfully.₩n");
}
```

데이터베이스에 새로운 에이전트 추가하는 상황이다.

먼저 visual studio의 console창에 띄워질 문구들을 작성한다.

MySQL에서의 쿼리문을 C++에 한 줄로 나타내기에는 길이가 상당한 관계로 나눠서 저장했다.

agent_name, agent_phone_number, agent_address의 정보를 받아서 agent에 정보를 저장하게 된다. 만약 저장이 정상적으로 되었다면 "New agent added successfuly"라는 문장이 뜬다.