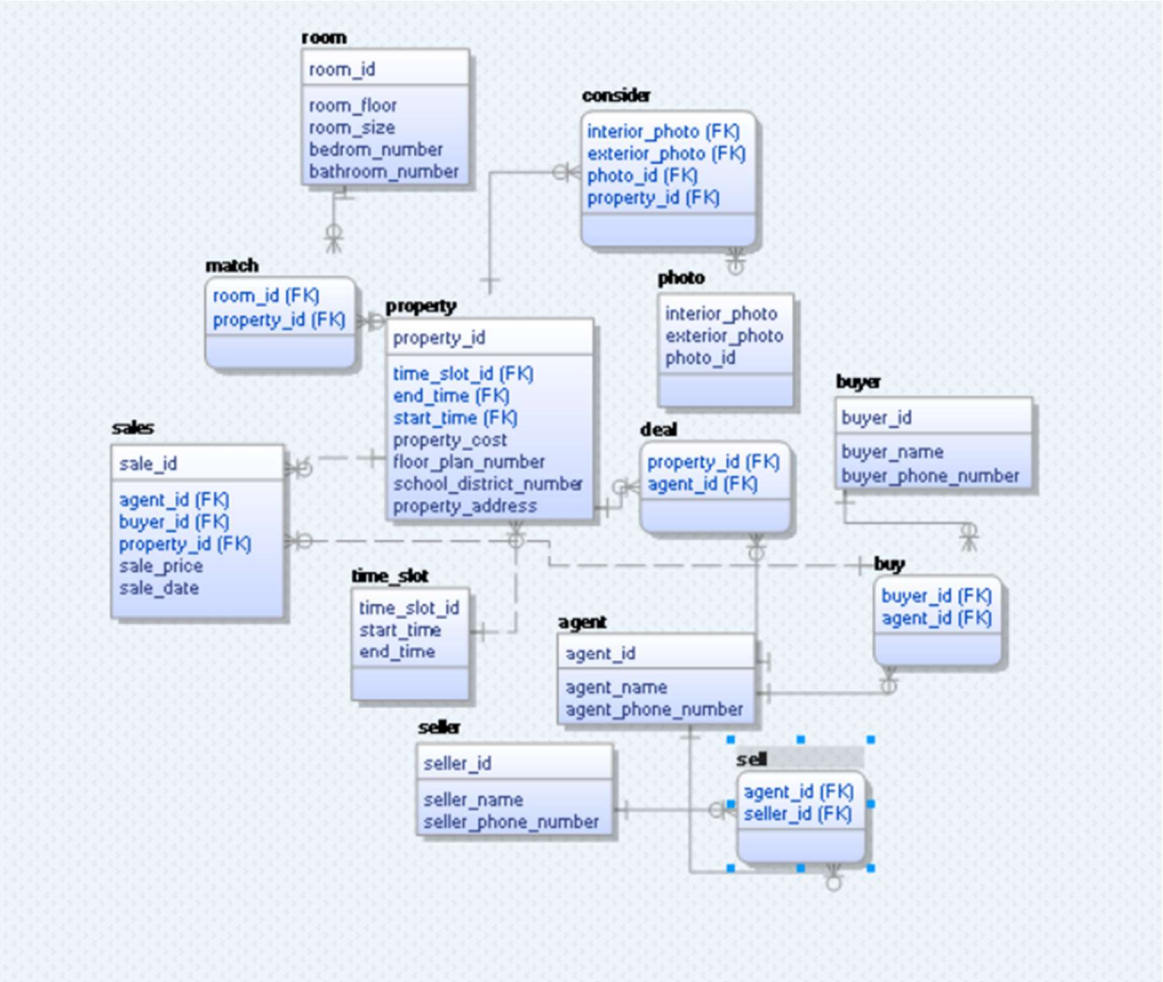
CSE4110 - Database System

Project1. E-R design and Relational Schema design

20212021 컴퓨터공학과 원대호

[1] BCNF DECOMPOSITION



위 diagram은 project1에서 완성된 E-R diagram이다. 각 entity별로 functional dependency를 확인했고, dependency preserving과 BCNF를 만족하는 지 여부를 확인했다.

[a] property

property(property_id, time_slot_id, end_time, start_time, property_cost, floor_plan_number, school_district_number, property_address)

property_id -> time_slot_id, end_time, start_time, property_cost, floor_plan_number, school_district_number, property_address

property 테이블은 BCNF를 만족한다. 그 이유는 모든 함수 종속성(property_id -> time_slot_id, end_time, start_time, property_cost, floor_plan_number, school_district_number, property_address)이

superkey인 property_id에 의해 결정되기 때문이다. BCNF는 모든 결정자가 candidate key 또는 superkey여야 한다는 조건을 갖추어야 한다. property table에서는 property_id가 모든 다른 속성을 결정할 수 있는 superkey로 작용하고 있다. 따라서 모든 함수 종속성이 이 superkey에 의해 결정된다. 이는 BCNF의 condition을 충족시키며, 따라서 property table은 BCNF를 만족한다.

[b] room

모든 함수 종속성(room_id -> room_floor, room_size, bedroom_number, bathroom_number)은 superkey인 room_id에 의해 결정된다. 이에 모든 결정자가 superkey이므로 room table은 BCNF를 만족한다.

[c] photo

모든 함수 종속성(photo_id -> interior_photo, exterior_photo)은 superkey인 photo_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. 따라서, photo table은 BCNF를 만족한다.

[d] time_slot

time_slot 테이블은 BCNF를 만족한다. 모든 함수 종속성(time_slot_id -> start_time, end_time)은 superkey인 time_slot_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. 따라서, 모든 결정자가 superkey인 조건을 충족하므로 time_slot table은 BCNF를 만족한다. 이로 인해 데이터 무결성과 일관성이 유지된다.

[e] agent

agent 테이블은 BCNF를 만족한다. 모든 함수 종속성(agent_id -> agent_name, agent_phone_number)은 superkey인 agent_id에 의해 결정된다. 이는 모든 결정자가 후보 키임을 의미한다. 따라서, 모든 결정자가 superkey인 조건을 충족하므로 agent table은 BCNF를 만족한다.

[f] seller

seller table은 BCNF를 만족한다. 모든 함수 종속성(seller_id -> seller_name, seller_phone_number)은 superkey인 seller_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. 따라서, 모든 결정자가 superkey인 조건을 충족하므로 seller table은 BCNF를 만족한다.

[g] match

match table은 BCNF를 만족한다. 모든 함수 종속성(room_id, property_id)은 복합 superkey인 room_id와 property_id에 의해 결정된다. 이는 복합 결정자가 superkey임을 의미한다. 따라서, 모든 결정자가 superkey인 조건을 충족하므로 match table은 BCNF를 만족한다. 이로 인해 데이터 무결성과 일관성이 유지된다.

[h] consier

consider table은 BCNF를 만족한다. 모든 함수 종속성(photo_id, property_id)은 복합 superkey인 photo_id와 property_id에 의해 결정된다. 따라서, 모든 결정자가 superkey인 조건을 충족하므로 consider table은 BCNF를 만족한다.

[i] deal

모든 함수 종속성(property_id, agent_id)은 복합 superkey인 property_id와 agent_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. 이에 deal 테이블은 BCNF를 만족한다.

[j] buy

buy 테이블은 BCNF를 만족한다. 모든 함수 종속성(buyer_id, agent_id)은 복합 superkey인 buyer_id와 agent_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. 따라서, 모든 결정자가 후보키인 조건을 충족하므로 buy 테이블은 BCNF를 만족합니다. 이는 데이터 무결성과 일관성을 유지하는 데 기여합니다.

[k] sell

모든 함수 종속성(agent_id, seller_id)은 복합 superkey인 agent_id와 seller_id에 의해 결정된다. 이는 모든 결정자가 superkey임을 의미한다. sell 테이블은 BCNF를 만족한다.

[ㅣ] buyer

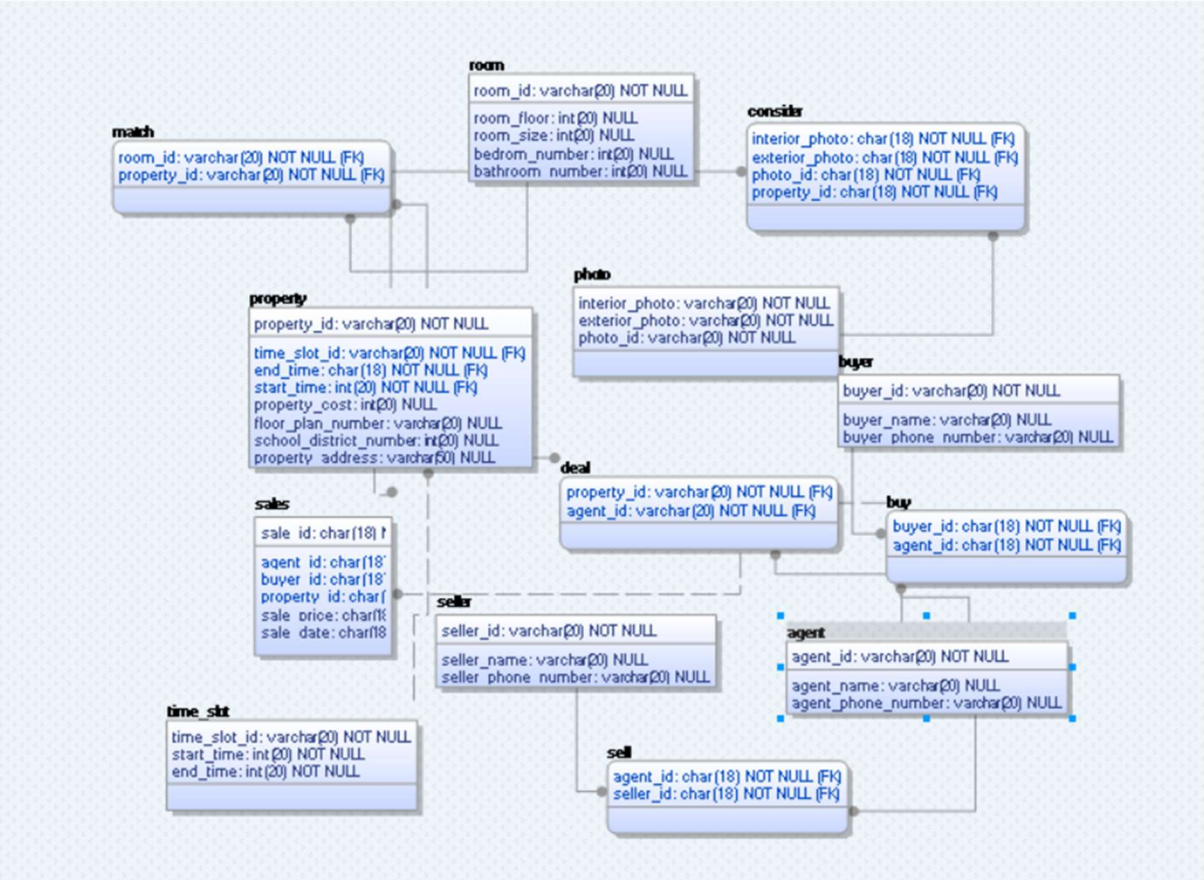
buyer table은 BCNF를 만족한다. 모든 함수 종속성(buyer_id -> buyer_name, buyer_phone_number)은 superkey인 buyer_id에 의해 결정된다. 이에 모든 결정자가 superkey인 조건을 충족하므로 buyer

테이블은 BCNF를 만족한다.

[j] sales

추가적으로 sales라는 table을 하나 더 생성했다. 이는 DB를 사용함에 있어 좀 더 효율적으로 physical diagram을 이용할 수 있을 것 같아 이러한 결정을 하게 됐다. Sales 테이블은 부동산 판매 거래 정보를 저장하는 테이블로, 각 판매 거래를 고유하게 식별하는 sale_id와 중개인(agent_id), 구매자(buyer_id), 부동산(property_id), 판매 가격(sale_price), 판매 날짜(sale_date) 필드를 포함한다. agent, buyer, property 테이블과 외래 키로 연결되어 있다. 각 판매 거래가 특정 중개인, 구매자, 부동산과 관련됨을 의미한다. 이를 통해 포괄적인 판매 거래 정보를 관리할 수 있다.

[2] physical diagram



다음은 [1]번에서 작성한 logical schema diagram을 physical schema diagram으로 변환한 것이다. 각 entity에 대한 설명은 아래와 같다.

room

room_id: 방 ID (기본 키)

room_floor: 방 층수

room_size: 방 크기

bedroom_number: 침실 수

bathroom_number: 욕실 수

match

room_id: 방 ID (외래 키, room 테이블 참조)

property_id: 부동산 ID (외래 키, property 테이블 참조)

property

property_id: 부동산 ID (기본 키)

time_slot_id: 시간 슬롯 ID (외래 키, time_slot 테이블 참조)

end_time: 종료 시간

start_time: 시작 시간

property_cost: 부동산 가격

floor_plan_number: 평면도 번호

school_district_number: 학군 번호

property_address: 부동산 주소

photo

photo_id: 사진 ID (기본 키)

interior_photo: 내부 사진

exterior_photo: 외부 사진

consider

property_id: 부동산 ID (외래 키, property 테이블 참조)

photo_id: 사진 ID (외래 키, photo 테이블 참조)

interior_photo: 내부 사진

exterior_photo: 외부 사진

sales

sale_id: 판매 ID (기본 키)

agent_id: 중개인 ID (외래 키, agent 테이블 참조)

buyer_id: 구매자 ID (외래 키, buyer 테이블 참조)

property_id: 부동산 ID (외래 키, property 테이블 참조)

sale_price: 판매 가격

sale_date: 판매 날짜

deal

property_id: 부동산 ID (외래 키, property 테이블 참조)

agent_id: 중개인 ID (외래 키, agent 테이블 참조)

time_slot

time_slot_id: 시간 슬롯 ID (기본 키)

start_time: 시작 시간

end_time: 종료 시간

seller

seller_id: 판매자 ID (기본 키)

seller_name: 판매자 이름

seller_phone_number: 판매자 전화번호

buyer

buyer_id: 구매자 ID (기본 키)

buyer_name: 구매자 이름

buyer_phone_number: 구매자 전화번호

agent

agent_id: 중개인 ID (기본 키)

agent_name: 중개인 이름

agent_phone_number: 중개인 전화번호

buy

buyer_id: 구매자 ID (외래 키, buyer 테이블 참조)

agent_id: 중개인 ID (외래 키, agent 테이블 참조)

sell

agent_id: 중개인 ID (외래 키, agent 테이블 참조)

seller_id: 판매자 ID (외래 키, seller 테이블 참조)

[3] quries

(TYPE1)

Find address of homes for sale in the district

"Mapo".

(TYPE1-1) Then find the costing between ₩

1,000,000,000 and ₩1,500,000,000.

```
// TYPE 1 쿼리 처리
void handleType1Queries(MYSQL* conn) {
    string query = "SELECT property_address FROM property WHERE property_address LIKE '%Mapo%';";
    executeAndDisplayQuery(conn, query, true);

    int choice;
    while (true) {
        cout << "₩n------- Subtypes in TYPE 1 -------₩n";
        cout << "       1. TYPE 1-1₩n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            query = "SELECT property_address FROM property WHERE property_cost BETWEEN 1000000000 AND 1500000000 AND property_address LIKE '%Mapo%';";
            executeAndDisplayQuery(conn, query, true);
            return;
        default:
            cout << "Invalid choice. Please try again.₩n";
        }
    }
}
```

```
---------- 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. Exit

Enter choice: 1
[property_address]
Seoul Mapo 452

------- Subtypes in TYPE 1 -------
       1. TYPE 1-1
Enter choice: 1
[property_address]
Seoul Mapo 452
```

이 함수는 'Mapo'가 포함된 주소의 부동산을 조회하며, 하위 메뉴에서 'Mapo' 지역의 가격이 10억 원에서 15억 원 사이인 부동산을 조회할 수 있다. 사용자가 옵션을 선택하면 해당 쿼리를 실행하고 결과를 표시한다. 잘못된 입력이 있을 경우 다시 입력을 요청한다. 출력으로 Seoul Mapo 452가 출력된다.

-(TYPE2)Find the address of homes for sale in the 8th

school district.

(TYPE2-1) Then find properties with 4 or more

bedrooms and 2 bathrooms.

```
// TYPE 2 쿼리 처리
void handleType2Queries(MYSQL* conn) {
    string query = "SELECT property_address FROM property WHERE school_district_number = 8;";
    executeAndDisplayQuery(conn, query, true);

    int choice;
    while (true) {
        cout << "₩n------- Subtypes in TYPE 2 -------₩n";
        cout << "       1. TYPE 2-1₩n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            query = "SELECT property_address FROM property WHERE property_address LIKE '%Seoul Jongno%';";
            executeAndDisplayQuery(conn, query, true);
            return;
        default:
            cout << "Invalid choice. Please try again.₩n";
        }
    }
}
```

```
---------- 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. Exit

Enter choice: 2
[property_address]
Seoul Mapo 489
Seoul Gangnam 567
Seoul Jongno 346

------- Subtypes in TYPE 2 -------
       1. TYPE 2-1
Enter choice: 1
[property_address]
Seoul Jongno 346
```

이 함수는 학군 번호가 8번인 부동산의 주소를 조회하며, 하위 메뉴에서 '서울 종로' 지역의 부동산을 조회할 수 있다. 사용자가 옵션을 선택하면 해당 쿼리를 실행하고 결과를 표시한다. 잘못된 입력이 있을 경우 다시 입력을 요청한다. 2를 mapo, Gangnam, jongno가 출력되고 1을 누르면 jongno가 출력된다.

(TYPE3)Find the name of the agent who has sold the most properties in the year 2022 by total won value.

(TYPE3-1) Then find the top k agents in the year 2023 by total won value.

(TYPE3-2) And then find the bottom 10% agents in the year 2021 by total won value.

```
Enter choice: 3
[agent_name]                    [TotalSales]
Big ho                          900000000

------- Subtypes in TYPE 3 -------
       1. TYPE 3-1
       2. TYPE 3-2

Enter choice: 1

------- TYPE 3-1 --------

** Find the top k agents in 2023 by total sales value. **
 Enter K: 3

[agent_name]                    [TotalSales]
da min                          950000000
uno ni                          850000000
```

```
------- Subtypes in TYPE 3 -------
       1. TYPE 3-1
       2. TYPE 3-2

 Enter choice: 2

[agent_name]                    [TotalSales]
da min                          1270000000
```

```
// TYPE 3 쿼리 처리
void handleType3Queries(MYSQL* conn) {
    string query = "SELECT a.agent_name, SUM(s.sale_price) AS TotalSales "
        "FROM agent a JOIN sales s ON a.agent_id = s.agent_id "
        "WHERE YEAR(s.sale_date) = 2022 "
        "GROUP BY a.agent_id "
        "ORDER BY TotalSales DESC "
        "LIMIT 1;";
    executeAndDisplayQuery(conn, query, true);

    int choice;
    while (true) {
        cout << "\n------- Subtypes in TYPE 3 -------\n";
        cout << "       1. TYPE 3-1\n";
        cout << "       2. TYPE 3-2\n\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1: {
            cout << "\n------- TYPE 3-1 --------\n";
            int k;
            cout << "\n** Find the top k agents in 2023 by total sales value. **\n";
            cout << " Enter K: ";
            cin >> k;
            cout << '\n';
            query = "SELECT a.agent_name, SUM(s.sale_price) AS TotalSales "
                "FROM agent a JOIN sales s ON a.agent_id = s.agent_id "
                "WHERE YEAR(s.sale_date) = 2023 "
                "GROUP BY a.agent_id "
                "ORDER BY TotalSales DESC "
                "LIMIT " + to_string(k) + ";";
            executeAndDisplayQuery(conn, query, true);
            return;
        }
        case 2: {
            cout << '\n';
            string countQuery = "SELECT ROUND(COUNT(DISTINCT a.agent_id) * 0.1) AS Bottom10PercentCount "
                "FROM agent a JOIN sales s ON a.agent_id = s.agent_id "
                "WHERE YEAR(s.sale_date) = 2021;";
            if (mysql_query(conn, countQuery.c_str())) {
                printf("Query Error: %s\n", mysql_error(conn));
                return;
            }
            MYSQL_RES* result = mysql_store_result(conn);
            if (!result) {
                printf("Result Error: %s\n", mysql_error(conn));
                return;
```

```
            }
            MYSQL_RES* result = mysql_store_result(conn);
            if (!result) {
                printf("Result Error: %s\n", mysql_error(conn));
                return;
            }
            MYSQL_ROW row = mysql_fetch_row(result);
            if (!row) {
                printf("Fetch Error: %s\n", mysql_error(conn));
                mysql_free_result(result);
                return;
            }
            int bottom10PercentCount = stoi(row[0] ? row[0] : "0");
            mysql_free_result(result);

            query = "SELECT agent_name, SUM(s.sale_price) AS TotalSales "
                "FROM agent a JOIN sales s ON a.agent_id = s.agent_id "
                "WHERE YEAR(s.sale_date) = 2021 "
                "GROUP BY a.agent_id "
                "ORDER BY TotalSales ASC "
                "LIMIT " + to_string(bottom10PercentCount) + ";";
            executeAndDisplayQuery(conn, query, true);
            return;
        }
        default:
            cout << "Invalid choice. Please try again.\n";
        }
    }
}
```

이 함수는 2022년 판매 총액이 가장 높은 중개인을 조회하고, 하위 메뉴에서 2023년 상위 k명의 중개인 또는 2021년 하위 10% 중개인을 조회할 수 있다. 사용자가 option을 선택하면 해당 쿼리를 실행하고 결과를 표시한다. 잘못된 입력이 있을 경우 다시 input을 요청한다.

(TYPE4)For each agent, compute the average selling price of properties sold in 2022, and the average time the

property was on the market.

(TYPE4-1) Then compute the maximum selling price of properties sold in 2023 for each agent.

(TYPE4-2) And then compute the longest time the property was on the market for each agent.

```
Enter choice: 4
[agent_name]                    [AvgSalePrice]                  [AvgMarketTime]
di mth                          750000000.0000                  321.0000

------- Subtypes in TYPE 4 -------
       1. TYPE 4-1
       2. TYPE 4-2
Enter choice: 1
[agent_name]                    [MaxSalePrice]
da min                          950000000
uno ni                          850000000

------- Subtypes in TYPE 4 -------
       1. TYPE 4-1
       2. TYPE 4-2
Enter choice: 2
[agent_name]                    [LongestMarketTime]
da min                          690
di mth                          836
uno ni                          680
```

```
// TYPE 4 쿼리 처리
void handleType4Queries(MYSQL* conn) {
    string query = "SELECT a.agent_name, AVG(s.sale_price) AS AvgSalePrice, AVG(DATEDIFF(p.end_time, s.sale_date)) AS AvgMarketTi
        "FROM agent a JOIN deal d ON a.agent_id = d.agent_id "
        "JOIN property p ON d.property_id = p.property_id "
        "JOIN sales s ON p.property_id = s.property_id "
        "WHERE YEAR(s.sale_date) = 2022 "
        "GROUP BY a.agent_id;";
    executeAndDisplayQuery(conn, query, true);

    int choice;
    while (true) {
        cout << "₩n------- Subtypes in TYPE 4 -------₩n";
        cout << "       1. TYPE 4-1₩n";
        cout << "       2. TYPE 4-2₩n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            query = "SELECT a.agent_name, MAX(s.sale_price) AS MaxSalePrice "
                "FROM agent a JOIN deal d ON a.agent_id = d.agent_id "
                "JOIN property p ON d.property_id = p.property_id "
                "JOIN sales s ON p.property_id = s.property_id "
                "WHERE YEAR(s.sale_date) = 2023 "
                "GROUP BY a.agent_id;";
            executeAndDisplayQuery(conn, query, true);
            break;
        case 2:
            query = "SELECT a.agent_name, MAX(DATEDIFF(p.end_time, s.sale_date)) AS LongestMarketTime "
                "FROM agent a JOIN deal d ON a.agent_id = d.agent_id "
                "JOIN property p ON d.property_id = p.property_id "
                "JOIN sales s ON p.property_id = s.property_id "
                "GROUP BY a.agent_id;";
            executeAndDisplayQuery(conn, query, true);
            return;
        default:
            cout << "Invalid choice. Please try again.₩n";
        }
    }
}
```

이 함수는 2022년 중개인의 평균 판매 가격과 평균 시장 시간을 조회하고, 하위 메뉴에서 2023년 최고 판매 가격을 가진 중개인 또는 가장 긴 시장 시간을 가진 중개인을 조회할 수 있다. 사용자가 옵션을 선택하면 해당 쿼리를 실행하고 결과를 표시한다. 잘못된 입력이 있을 경우 다시 입력을 요청한다.

(TYPE5)Showphotos of the most expensive studio, one-bedroom, multi-bedroom apartment(s), and detached house(s), respectively, from the database.

```
// TYPE 5 쿼리 처리
void handleType5Queries(MYSQL* conn) {
    string query;

    query = "SELECT ph.photo_id, p.property_id, p.floor_plan_number, ph.interior_photo "
        "FROM photo ph JOIN property p ON ph.property_id = p.property_id "
        "WHERE p.floor_plan_number = 'studio' "
        "ORDER BY p.property_cost DESC LIMIT 1;";
    executeAndDisplayQuery(conn, query, true);

    query = "SELECT ph.photo_id, p.property_id, p.floor_plan_number, ph.interior_photo "
        "FROM photo ph JOIN property p ON ph.property_id = p.property_id "
        "WHERE p.floor_plan_number = 'one-bedroom' "
        "ORDER BY p.property_cost DESC LIMIT 1;";
    executeAndDisplayQuery(conn, query, false);

    query = "SELECT ph.photo_id, p.property_id, p.floor_plan_number, ph.interior_photo "
        "FROM photo ph JOIN property p ON ph.property_id = p.property_id "
        "WHERE p.floor_plan_number = 'multi-bedroom' "
        "ORDER BY p.property_cost DESC LIMIT 1;";
    executeAndDisplayQuery(conn, query, false);

    query = "SELECT ph.photo_id, p.property_id, p.floor_plan_number, ph.interior_photo "
        "FROM photo ph JOIN property p ON ph.property_id = p.property_id "
        "WHERE p.floor_plan_number = 'detached' "
        "ORDER BY p.property_cost DESC LIMIT 1;";
    executeAndDisplayQuery(conn, query, false);
}
```

```
---------- 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. Exit

Enter choice: 5
[photo_id]                      [property_id]               [floor_plan_number]         [interior_photo]

PB001                           P001                        studio                      Interior

PB007                           P004                        detached                    Exterior
```

이 함수는 새로운 판매 기록을 입력받아 sales 테이블에 삽입하는 역할을 한다. 먼저 사용자로부터 판매 ID, 부동산 ID, 판매 가격, 판매 날짜, 구매자 ID, 중개인 ID를 입력받는다. 각 입력값을 차례로 콘솔에서 받아 변수에 저장한다. 이 data를 사용하여 SQL INSERT 쿼리를 구성한다. 쿼리는 sales 테이블에 새로운 행을 삽입하도록 만들어진다. 구성된 쿼리를 executeAndDisplayQuery 함수를 통해 실행한다. 이를 통해 새로운 판매 record가 데이터베이스에 저장된다.

(TYPE6)Record the sale of a property that had been listed as being available. This entails storing the sales price, the buyer, the selling agent, the buyer's agent(if any), and the 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. Exit

Enter choice: 6

TYPE 6 - Record a sale:
Enter SaleID: SA016
Enter PropertyID: P009
Enter SalePrice: 9000000000
Enter SaleDate (YYYY-MM-DD): 2021-12-31
Enter BuyerID: B003
Enter AgentID: A001
```

```
// TYPE 6 쿼리 처리
void handleType6Queries(MYSQL* conn) {
    cout << "\nTYPE 6 - Record a sale:\n";
    string saleID, propertyID, salePrice, saleDate, buyerID, agentID;
    cout << "Enter SaleID: ";
    cin >> saleID;
    cout << "Enter PropertyID: ";
    cin >> propertyID;
    cout << "Enter SalePrice: ";
    cin >> salePrice;
    cout << "Enter SaleDate (YYYY-MM-DD): ";
    cin >> saleDate;
    cout << "Enter BuyerID: ";
    cin >> buyerID;
    cout << "Enter AgentID: ";
    cin >> agentID;
    string query = "INSERT INTO sales (sale_id, property_id, sale_price, sale_date, buyer_id, agent_id) "
        "VALUES ('" + saleID + "', '" + propertyID + "', '" + salePrice + "', '" + saleDate + "', '" + buyerID + "', '" + agentID + "');";
    executeAndDisplayQuery(conn, query, true);
}
```

이 함수는 새로운 판매 기록을 input으로 받아 sales table에 삽입한다. 사용자로부터 판매 ID, 부동산 ID, 판매 가격, 판매 날짜, 구매자 ID, 중개인 ID를 input으로 받는다. 입력받은 값을 사용해 SQL INSERT 쿼리를 구성한다. 구성된 쿼리를 실행하여 데이터베이스에 저장한다.

(TYPE7) Addanewagentto the database.

```
---------- 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. Exit

Enter choice: 7

TYPE 7 - Add a new agent:
Enter AgentID: A006
Enter AgentName: dksk dsa
Enter PhoneNumber: 010-1231-2141
```

```
// TYPE 7 쿼리 처리
void handleType7Queries(MYSQL* conn) {
    cout << "\nTYPE 7 - Add a new agent:\n";
    string agentID, agentName, phoneNumber;
    cout << "Enter AgentID: ";
    cin >> agentID;
    cout << "Enter AgentName: ";
    cin.ignore();
    getline(cin, agentName);
    cout << "Enter PhoneNumber: ";
    cin >> phoneNumber;
    string query = "INSERT INTO agent (agent_id, agent_name, agent_phone_number) "
        "VALUES ('" + agentID + "', '" + agentName + "', '" + phoneNumber + "');";
    executeAndDisplayQuery(conn, query, true);
}
```

이 코드는 MySQL database에 새로운 agent를 추가하는 함수이다. 사용자에게 에이전트 ID, 이름, 전화번호를 input으로 받는다. 입력된 정보를 기반으로 SQL INSERT 쿼리를 작성한다. 작성된 query를 실행하여 데이터베이스에 새로운 agent를 추가한다.