

DATABASE SYSTEM

Project 2

Normalization and Query Processing

**Automobile Company**

컴퓨터공학과

20181668

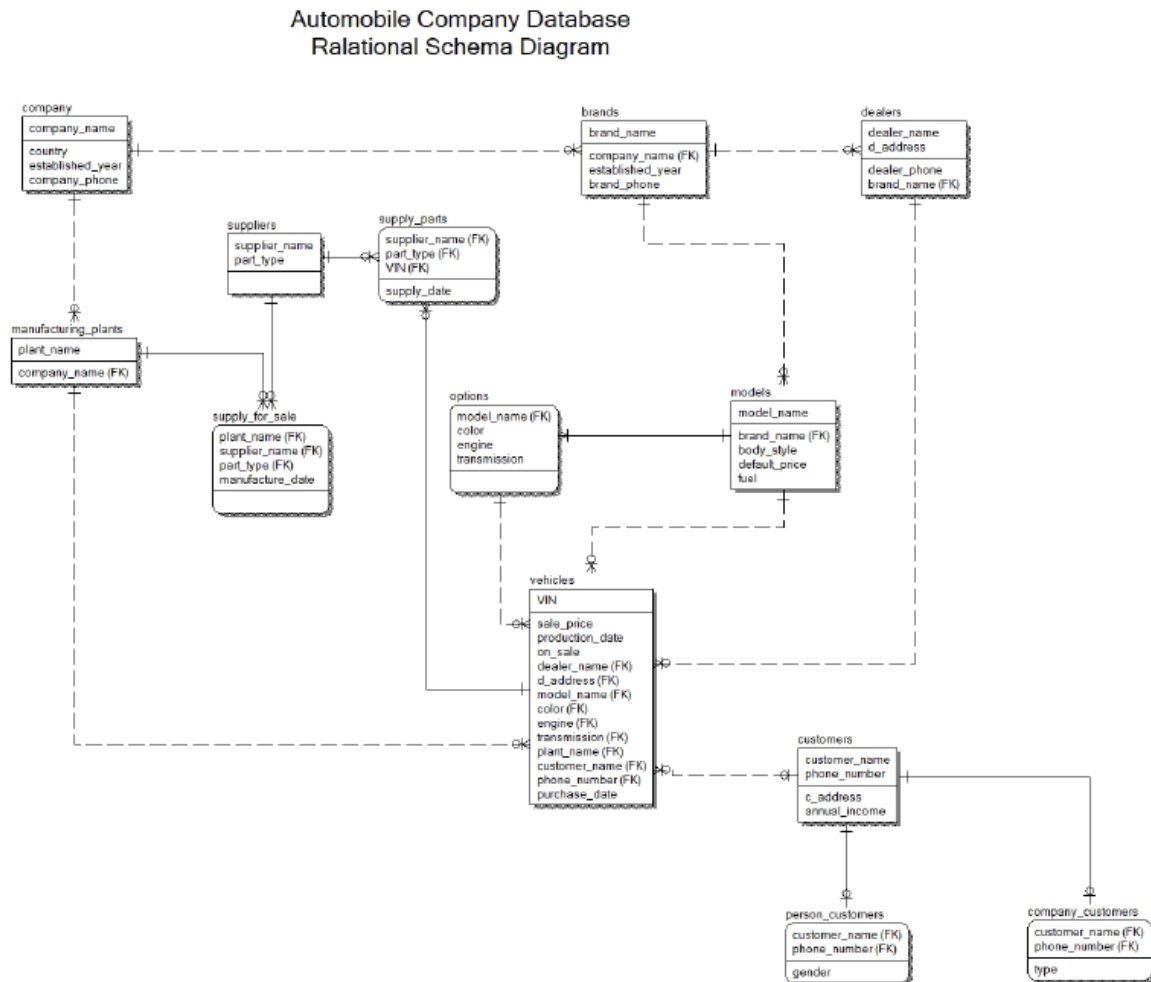
이예진

# 목차

1. BCNF Decomposition
2. Physical Schema Diagram
3. ODBC C language code
  - 3.1. C code description
  - 3.2. Query result

# 1. BCNF Decomposition

각 릴레이션이 BCNF인지 확인하고 BCNF가 아니라면 decomposition을 한 결과 아래와 같은 decomposed logical schema diagram이 생성되었다.



교재의 Chapter 7에서 속성의 값이 더 작은 부분으로 나누어지면 원자성을 갖지 않는다고 나와 있다. 원자성을 위해 기존에 작성했던 릴레이션 스키마를 수정했다. part\_type을 e\_ge\_210414와 같이 제조사와 날자를 포함하는 기존 형태에서 part\_type에 engine 또는 transmission인 값만 갖도록 하였다. supply\_for\_sale에서 PK에 manufacture\_date를 추가했다. 그리고 기존에 supply\_parts는 options와 연결되어 있었지만 options가 아닌 vehicles와 연결되도록 수정하였다. 그리고 vehicles 테이블을 작성할 때 plant\_type에 assemble이라는 값이 중복으로 저장되는 문제를 해결하기 위해 plant\_name에 type에 대한 정보를 포함하고 있다고 하고 데이터베이스를 설계하였다.

아래는 각 테이블에 BCNF인지 확인하는 과정이다.

(1) company

-Functional Dependency : company\_name -> country, established\_year, company\_phone

회사 이름에 대한 값이 동일하면 국가, 설립연도, 회사 전화번호에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. company\_name이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. company\_name의 closure는 릴레이션의 모든 속성을 포함한다.

## (2) brands

-Functional Dependency : brand\_name -> company\_name, established\_year, brand\_phone

브랜드 이름에 대한 값이 동일하면 회사 이름, 설립연도, 브랜드 전화번호에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. brand\_name이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. brand\_name의 closure는 릴레이션의 모든 속성을 포함한다.

## (3) dealers

-Functional Dependency : dealer\_name, d\_address -> dealer\_phone, brand\_name

딜러점 이름, 딜러 주소에 대한 값이 동일하면 딜러 전화번호, 브랜드 이름에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. dealer\_name, d\_address가 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. dealer\_name, d\_address의 closure는 릴레이션의 모든 속성을 포함한다.

## (4) models

-Functional Dependency : model\_name -> brand\_name, body\_style, default\_price, fuel

모델 이름에 대한 값이 동일하면 브랜드 이름, 외형, 기본 가격, 연료에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. model\_name이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. model\_name의 closure는 릴레이션의 모든 속성을 포함한다.

## (5) options

-Functional Dependency : model\_name, color, engine, transmission -> model\_name, color, engine, transmission

모델 이름, 색, 엔진 변속기 값이 테이블의 값들을 식별한다. 결정자와 종속자가 같은  $a \rightarrow a$  형태의 함수 종속성이다. Functional Dependency가 trivial이기 때문에 조건을 만족하므로 BCNF를 만족한다. 결정자와 종속자가 같아서 종속자가 결정자의 부분집합이다.

## (6) vehicles

-Functional Dependency : VIN -> sale\_price, production\_date, on\_sale, dealer\_name, d\_address, model\_name, color, engine, transmission, plant\_name, customer\_name, phone\_number, purchase\_date

VIN에 대한 값이 동일하면 판매 가격, 생산 날짜, 판매여부, 딜러점 이름, 딜러 주소, 모델 이름, 색상, 엔진, 변속기, 공장 이름, 고객 이름, 고객 전화번호, 구매 날짜도 동일하다. 그러므로 함수적 종속성이 있다.

VIN이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. VIN의 closure는 릴레이션의 모든 속성을 포함한다.

## (7) supply\_parts

-Functional Dependency : supplier\_name, part\_type, VIN -> supply\_date

공급점의 이름과, 공급한 부품, VIN에 대한 값이 동일하면 공급 날짜도 동일하다. 그러므로 함수적 종속성이 있다. supplier\_name, part\_type, VIN이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. supplier\_name, part\_type, VIN의 closure는 릴레이션의 모든 속성을 포함한다.

## (8) suppliers

-Functional Dependency : supplier\_name, part\_type -> supplier\_name, part\_type

공급점의 이름, 공급한 부품이 테이블의 값들을 식별한다. 결정자와 종속자가 같은  $a \rightarrow a$  형태의 함수 종속성이다. Functional Dependency가 trivial이기 때문에 조건을 만족하므로 BCNF를 만족한다. 결정자와 종속자가 같아서 종속자가 결정자의 부분집합이다.

(9) supply\_for\_sale

-Functional Dependency : plant\_name, supplier\_name, part\_type, manufacture\_date ->

plant\_name, supplier\_name, part\_type, manufacture\_date

공장의 이름, 공급점의 이름, 공급한 부품, 제조 날짜가 테이블의 값들을 식별한다. 결정자와 종속자가 같은  $a \rightarrow a$  형태의 함수 종속성이다. Functional Dependency가 trivial이기 때문에 조건을 만족하므로 BCNF를 만족한다. 결정자와 종속자가 같아서 종속자가 결정자의 부분집합이다.

(10) manufacturing\_plants

-Functional Dependency : plant\_name -> company\_name

공장의 이름이 동일하면 공장이 소속된 회사에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. plant\_name이 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. plant\_name의 closure는 릴레이션의 모든 속성을 포함한다.

(11) customers

-Functional Dependency : customer\_name, phone\_number -> c\_address, annual\_income

고객의 이름과, 전화번호 속성에 대한 값이 동일하면 고객의 주소와 연수입에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. customer\_name, phone\_number가 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. customer\_name, phone\_number의 closure는 릴레이션의 모든 속성을 포함한다.

(12) person\_customers

-Functional Dependency : customer\_name, phone\_number -> gender

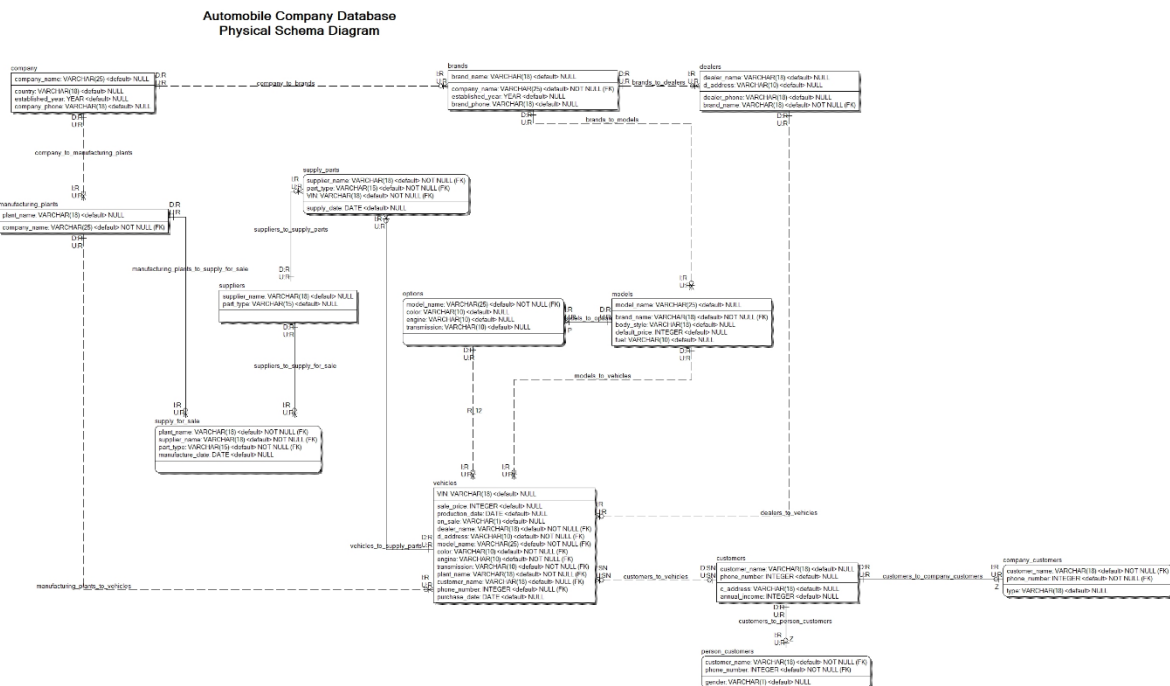
고객의 이름과, 전화번호 속성에 대한 값이 동일하면 고객의 성별에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. customer\_name, phone\_number가 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. customer\_name, phone\_number의 closure는 릴레이션의 모든 속성을 포함한다.

(13) company\_customers

-Functional Dependency : customer\_name, phone\_number -> type

고객의 이름과, 전화번호 속성에 대한 값이 동일하면 고객의 기업 종류에 대한 값도 동일하다. 그러므로 함수적 종속성이 있다. customer\_name, phone\_number가 primary key이므로 super key이고 조건을 만족하므로 BCNF를 만족한다. customer\_name, phone\_number의 closure는 릴레이션의 모든 속성을 포함한다.

## 2. Physical Schema Diagram



위의 그림은 1에서 만든 decomposed logical schema diagram을 바탕으로 생성한 physical diagram이다. 데이터 타입, 도메인, 제약 조건, 관계 타입, NULL에 대한 정보를 포함한다.

### - Entities

#### (1) company

회사 이름은 최대 25개 문자, 국가는 최대 18개 문자, 설립 연도는 연도 자료형, 회사 전화번호는 정수 자료형으로 설정하였다. 전화번호는 "-" 문자 없이 "12345678" 같은 형태로 저장된다. 다른 릴레이션들도 동일한 방식을 사용한다. 모든 속성의 도메인은 default로 설정하였고 NULL을 허용한다.

#### (2) brands

브랜드 이름은 최대 18개 문자, 회사 이름은 최대 25개 문자, 설립 연도는 연도 자료형, 브랜드 전화번호는 정수 자료형으로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 company\_name은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (3) dealers

딜러점 이름은 최대 18개 문자, 딜러점 주소는 최대 10개 문자, 딜러점 전화번호는 정수 자료형, 브랜드 이름은 최대 18개 문자로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 brand\_name은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (4) models

모델 이름은 최대 25개 문자, 브랜드 이름은 최대 18개 문자, 외형은 최대 18개 문자, 기본 가격은 정수 자료형, 연료는 최대 10개 문자로 설정하였다. 가격은 "5000" 같은 형태로 저장되며 단위는 10달러이다. 즉 "5000"은 50000달러이고 약 5000만원이다. 모든 속성의 도메인은 default로 설정하였고 brand\_name은 NOT

NULL이고 나머지는 NULL을 허용한다.

#### (5) options

모델 이름은 최대 25개 문자, 색상은 최대 10개 문자, 엔진은 최대 10개 문자, 변속기는 최대 10개 문자로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 model\_name은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (6) vehicles

VIN은 최대 18개 문자, 판매 가격은 정수 자료형, 생산 날짜는 날짜 자료형, 판매 여부는 최대 1개 문자, 딜러점 이름은 최대 18개 문자, 딜러점 주소는 최대 10개 문자, 모델 이름은 최대 25개 문자, 색상은 최대 10개 문자, 엔진은 최대 10개 문자, 변속기는 최대 10개 문자, 공장 이름은 최대 18개 문자, 고객 이름은 최대 18개 문자, 전화번호는 정수 자료형, 구입 날짜는 날짜 자료형으로 설정하였다. VIN은 실제와 동일한 형식을 사용하여 저장하였다. 판매 여부는 팔린 자동차의 경우 "N"이 저장되고 판매 중인 경우 "Y"가 저장된다.

모든 속성의 도메인은 default로 설정하였고 FK 중에 customer\_name과 phone\_number를 제외한 나머지 FK는 모두 NOT NULL이다. 전체에서 NOT NULL을 제외한 나머지는 NULL을 허용한다. customer의 경우에는 아직 팔리지 않은 자동차이면 정보가 없으므로 NULL을 허용하는 것이다.

#### (7) supply\_parts

공급점 이름은 최대 18개 문자, 부품 종류는 최대 15개 문자, VIN은 최대 18개 문자, 공급 날짜는 날짜 자료형으로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 FK들은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (8) suppliers

공급점 이름은 최대 18개 문자, 부품 종류는 최대 15개 문자로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 모두 NULL을 허용한다.

#### (9) supply\_for\_sale

공장 이름은 최대 18개 문자, 공급점 이름은 최대 18개 문자, 부품 종류는 최대 15개 문자, 제조 날짜는 날짜 자료형으로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 FK들은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (10) manufacturing\_plants

공장 이름은 최대 18개 문자, 회사 이름은 최대 25개 문자로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 FK들은 NOT NULL이고 나머지는 NULL을 허용한다.

#### (11) customers

고객 이름은 최대 18개 문자, 전화번호는 정수 자료형, 고객 주소는 최대 18개 문자, 연수입은 정수 자료형으로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 모두 NULL을 허용한다.

#### (12) person\_customers

고객 이름은 최대 18개 문자, 전화번호는 정수 자료형, 성별은 최대 1개의 문자로 설정하였다. 성별은 남자인 경우 "M", 여자인 경우 "F"가 저장된다. 모든 속성의 도메인은 default로 설정하였고 FK들은 NOT NULL이고 나머지는 NULL을 허용한다.

### (13) company\_customers

고객 이름은 최대 18개 문자, 전화번호는 정수 자료형, 회사 종료는 최대 18개 문자로 설정하였다. 모든 속성의 도메인은 default로 설정하였고 FK들은 NOT NULL이고 나머지는 NULL을 허용한다.

#### - Relationships

FK Constraint : brands\_to\_dealers, brands\_to\_models, company\_to\_brands, company\_to\_manufacturing\_plants, customers\_to\_company\_customers, customers\_to\_person\_customers, customers\_to\_vehicles, dealers\_to\_vehicles, manufacturing\_plants\_to\_supply\_for\_sale, manufacturing\_plants\_to\_vehicles, models\_to\_options, models\_to\_vehicles, vehicles\_to\_supply\_parts, options\_to\_vehicles, suppliers\_to\_supply\_for\_sale, suppliers\_to\_supply\_parts

위의 제약들은 모두 프로젝트 1과 동일하다. 내용이 중복되므로 따로 작성하지 않았다.

## 3. ODBC C language code

### 3.1. C code description

SQL을 중심으로 중요한 부분만 코드를 분석하였다. 다른 부분 설명은 cpp파일에 주석을 달았다.

```
// MySQL 에 접속할 때 필요한 정보이다.
const char* host = "localhost";
const char* user = "root";
const char* pw = "yj0616!";
const char* db = "prj2";
// MySQL 에 연결하여 사용할 변수들을 정의한다.
MYSQL* connection = NULL;
MYSQL conn;
MYSQL_RES* sql_result;
MYSQL_ROW sql_row; // 쿼리 결과를 row 1 개씩 저장한다.

type 1
// 브랜드 이름, 자동차 개수, 총 판매 가격, 구매한 연도
sprintf(q1, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), YEAR(v.purchase_date) AS pyr
FROM vehicles AS v // vehicle, models, brands 를 조인한다.
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
WHERE v.on_sale = \"N\" and b.brand_name = \"%s\" // on_sale 이 N 이면 이미 팔렸다는 의미이다.
and (2021 - YEAR(v.purchase_date) > 0) and (2021 - YEAR(v.purchase_date) <= %d) // 2021 년 기준으로 계산한다.
GROUP BY pyr", b, k); // 연도를 기준으로 그룹화한다.

type 1-1
// 브랜드 이름, 자동차 개수, 총 판매액, 구매 연도, 손님 성별
sprintf(q11, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), YEAR(v.purchase_date) AS pyr, pc.gender
FROM vehicles AS v // vehicles, models, brands, 사람 손님의 대한 정보를 조인한다.
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
JOIN person_customers AS pc ON v.customer_name = pc.customer_name AND v.phone_number = pc.phone_number
WHERE v.on_sale = \"N\" and b.brand_name = \"%s\" // 위에 나왔던 내용과 같다.
and (2021 - YEAR(v.purchase_date) > 0) and (2021 - YEAR(v.purchase_date) <= %d)
```



```
GROUP BY pyr, pc.gender", b, k); // 구매 연도와 성별로 그룹화한다.
```

type 1-1-1

```
sprintf(q111, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), YEAR(v.purchase_date) AS pyr, pc.gender,
c.annual_income // 브랜드 이름, 자동차 개수, 총 판매액, 구매 연도, 손님 성별, 손님의 연수입
FROM vehicles AS v // vehicles, models, brands, 사람 손님(성별 때문에), 손님(연수입 때문에) 정보를 조인한다.
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
JOIN person_customers AS pc ON v.customer_name = pc.customer_name AND v.phone_number = pc.phone_number
JOIN customers AS c ON v.customer_name = c.customer_name AND v.phone_number = c.phone_number
WHERE v.on_sale = \"N\" and b.brand_name = \"%s\" // 위에 나왔던 것과 같다.
and (2021 - YEAR(v.purchase_date) > 0) and (2021 - YEAR(v.purchase_date) <= %d)
GROUP BY pyr, pc.gender, c.annual_income // 구매 연도, 성별, 연수입으로 그룹화한다.
ORDER BY pyr, c.annual_income DESC", b, k);
```

type 2

```
// 브랜드 이름, 자동차 개수, 총 판매액, 구매 월
sprintf(q2, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), MONTH(v.purchase_date) AS pm
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
WHERE (YEAR(v.purchase_date) = 2021) AND v.on_sale = \"N\" // 2021 년을 기준으로 월을 계산한다.
and (6 - MONTH(v.purchase_date) > 0) and (6 - MONTH(v.purchase_date) <= %d) // 6 월을 기준으로 차이를 구한다.
GROUP BY b.brand_name, pm ORDER BY pm", k); // 브랜드 이름, 월로 그룹화한다.
```

type 2-1

```
// 브랜드 이름, 판매 자동차 개수, 총 판매액, 구매 월, 손님 성별
sprintf(q21, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), MONTH(v.purchase_date) AS pm, pc.gender
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
JOIN person_customers AS pc ON v.customer_name = pc.customer_name AND v.phone_number = pc.phone_number
WHERE YEAR(v.purchase_date) = 2021 AND v.on_sale = \"N\"
and (6 - MONTH(v.purchase_date) > 0) and (6 - MONTH(v.purchase_date) <= %d)
GROUP BY b.brand_name, pm, pc.gender ORDER BY pm", k); // 브랜드 이름과 월로 그룹화한다.
```

type 2-1-1

```
sprintf(q211, "SELECT b.brand_name, COUNT(v.VIN), SUM(v.sale_price), MONTH(v.purchase_date) AS pm, pc.gender,
c.annual_income // 브랜드 이름, 판매 자동차 개수, 총 판매액, 월, 성별, 연수입
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
JOIN customers AS c ON v.customer_name = c.customer_name AND v.phone_number = c.phone_number
JOIN person_customers AS pc ON v.customer_name = pc.customer_name AND v.phone_number = pc.phone_number
WHERE YEAR(v.purchase_date) = 2021 AND v.on_sale = \"N\"
and (6 - MONTH(v.purchase_date) > 0) and (6 - MONTH(v.purchase_date) <= %d)
GROUP BY b.brand_name, pm, pc.gender, c.annual_income // 브랜드 이름, 월, 연수입으로 그룹화한다.
ORDER BY pm, c.annual_income DESC", k);
```

type 3

```
sprintf(q3, "SELECT * // 모든 컬럼
FROM supply_parts
WHERE supplier_name = \"%s\" AND part_type = \"transmission\" // 입력받은 공급점이고 부품 중 변속기만
AND supply_date between DATE('%s') AND DATE('%s')", sup_name, fd, sd); // 입력 받은 날짜 내에 존재
```

type 3 (참고용 테이블 출력)

```

sprintf(q33, "SELECT *
FROM supply_for_sale
WHERE supplier_name = \"%s\" AND part_type = \"transmission\"
AND manufacture_date <= DATE('%s')", sup_name, fd); // 해당 날짜 전에 공급했던 내역을 모두 보고자 한다.

type 3-1
// 차량 번호, 고객 이름, 고객 전화번호, 공급한 날짜, 구매 날짜
sprintf(q31, "SELECT sp.VIN, v.customer_name, v.phone_number, sp.supply_date, v.purchase_date
FROM supply_parts AS sp
JOIN vehicles AS v ON v.VIN = sp.VIN
WHERE sp.supplier_name = \"%s\" AND part_type = \"transmission\"
AND v.on_sale = \"N\" // 판매 완료된 차량들 중에 찾기 위한 조건
AND sp.supply_date between DATE('%s') AND DATE('%s')", sup_name, fd, sd);

type 3-2
// 딜러점 이름, 딜러점 주소, 차량 번호, 변속기, 공급한 날짜, 구매 날짜
sprintf(q32, "SELECT v.dealer_name, v.d_address, sp.VIN, o.transmission, sp.supply_date, v.purchase_date
FROM vehicles AS v
JOIN supply_parts AS sp ON v.VIN = sp.VIN
// 변속기 종류를 찾기 위해 options 테이블과 조인한다.
JOIN options AS o ON o.model_name = v.model_name AND o.color = v.color AND o.engine = v.engine AND
o.transmission = v.transmission
WHERE sp.supplier_name = \"%s\" AND part_type = \"transmission\"
AND v.on_sale = \"N\" AND sp.supply_date between DATE('%s') AND DATE('%s')", sup_name, fd, sd);

type 4
sprintf(q4, "SELECT b.brand_name, SUM(v.sale_price) // 브랜드 이름, 총 판매액
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
WHERE v.on_sale = \"N\" AND YEAR(v.purchase_date) = %d // 이미 판매 완료된 자동차이고 구매 연도는 입력받은 값
GROUP BY b.brand_name
ORDER BY SUM(v.sale_price) DESC LIMIT %d", y, k); // 총 판매액 정렬해서 입력받은 개수만큼 상위 항목 출력

type 5
sprintf(q5, "SELECT b.brand_name, COUNT(v.VIN) AS unit // 브랜드 이름, 판매된 차량 개수
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
JOIN brands AS b ON m.brand_name = b.brand_name
WHERE v.on_sale = \"N\" AND YEAR(v.purchase_date) = %d // 판매 완료된 차량이고 구매 연도는 입력받은 값
GROUP BY b.brand_name // 브랜드 이름으로 그룹화
ORDER BY COUNT(v.VIN) DESC LIMIT %d", y, k); // 판매 차량 개수로 정렬하여 상위 항목 출력 개수는 입력받은만큼

type 6
sprintf(q6, "SELECT MONTH(v.purchase_date) as pc_month, COUNT(VIN) // 구매 월, 판매된 자동차 개수
FROM vehicles AS v
JOIN models AS m ON v.model_name = m.model_name
WHERE m.body_style = \"Convertible\" AND v.on_sale = \"N\" // 차량 외형이 컨버터블이고 이미 판매 완료된 차량
GROUP BY pc_month ORDER BY COUNT(VIN) DESC"); // 월로 그룹화하고 판매 자동차 개수로 정렬

type 7
// 딜러 이름, 딜러 주소, inventory 에 있었던 일수
sprintf(q7, "SELECT dealer_name, d_address, AVG(timestampdiff(DAY, production_date, CURDATE())) as avg_date
FROM vehicles
WHERE on_sale = \"Y\" // 판매 중인 차량
GROUP BY dealer_name, d_address ORDER BY avg_date DESC"); // 딜러 정보로 그룹화하고 평균 기간 긴 순서로 정렬

```

```
// CRUD 파일 사용한다.
FILE* fp = fopen("20181668.txt", "r");
fp = fopen("20181668_2.txt", "r");
```

### 3.2. Query result

쿼리 실행 결과에 대한 내용이다. 처음 실행을 하면 쿼리 타입을 고를 수 있다. 0을 입력하면 프로그램을 종료한다. 쿼리나 서브쿼리를 나가고 싶으면 0을 입력하면 된다. 0을 입력하지 않으면 질의가 반복되고 0을 입력하면 그 바로 이전 단계로 간다.

```
Connection Succeed
----- 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
1
----- TYPE 1 -----
** Show the sales trends for a particular brand over the past k years. **
Which brand? : Audi
Which K? : 3
=====
brand_name    unit    dollar    year
=====
Audi          3       20200     2018
Audi          5       56800     2019
Audi          4       33800     2020
=====
----- TYPE 1-1 -----
** Break this data out by gender of the buyer **
** 0 : skip **
** 1 : show results **
Input : 1
=====
brand_name    unit    dollar    year    gender
=====
Audi          2       15200     2018     F
Audi          1       5000      2018     M
Audi          1       7600      2019     F
Audi          2       33200     2019     M
Audi          2       15200     2020     F
Audi          1       7600      2020     M
=====
----- TYPE 1-1-1 -----
** Break this data out by gender of the buyer and then break out by income **
** 0 : skip **
** 1 : show results **
Input : 1
=====
brand_name    unit    dollar    year    gender    income
=====
Audi          1       5000      2018     M       10000
Audi          1       7600      2018     F       7000
Audi          1       7600      2018     F       6000
Audi          1       25600     2019     M       8000
Audi          1       7600      2019     F       6000
Audi          1       7600      2019     M       4000
Audi          1       7600      2020     F       7000
Audi          1       7600      2020     F       6000
Audi          1       7600      2020     M       4000
=====
```

(TYPE 1)

타입 1을 선택하면 브랜드 이름을 입력 받고 지난 몇 년간의 트렌드를 볼 것인 것 입력 받는다. 위의 예제에서는 아우디 브랜드의 지난 3년간 판매 트렌드를 보여준다. 2021년을 기준으로 3년인 2018, 2019, 2020년의 판매 자동차 수(유닛)와 판매액(달러)를 표로 보여준다. 연도는 오름차순으로 정렬되어 출력된다.

(TYPE 1-1)

타입 1의 결과가 끝나면 1-1을 볼 것인지 스킵할 것인지 입력해야 한다. 스킵을 원하면 0을, 결과를 보려면 1을 입력한다. 결과를 보기위해 0이 아닌 다른 숫자를 입력해도 되지만 숫자만 입력이 가능하다. 1을 입력하면 타입 1의 결과를 성별로 그룹화하여 결과를 보여준다. 2018년, 2019년, 2020년의 각 판매 자동차 수와 판매액을 성별로 나누어 보여준다. 그룹화가 (연도 + 성별)로 되었다고 보면 된다. 연도는 오름차순으로 정렬되어 출력된다. which k? 옆에 0을 입력하면 쿼리를 나갈 수 있다고 추가했다. 기업 고객은 성별이 없으므로 사람 고객들을 대상으로 쿼리를 진행한 것이다. 타입 2-1에서도 마찬가지이다.

(TYPE 1-1-1)

타입 1-1의 결과가 끝나면 1-1-1을 볼 것인지 스킵할 것인지 입력해야 한다. 스킵을 원하면 0을, 결과를 보려면 1을 입력한다. 결과를 보기위해 0이 아닌 다른 숫자를 입력해도 되지만 숫자만 입력이 가능하다. 1을 입력하면 타입 1-1의 결과를 연간 수입으로 그룹화하여 결과를 보여준다. 그룹화가 (연도 + 성별 + 연수입)로 되었다고 보면 된다. 수입과 판매액은 단위가 100달러이다. 예를 들어 10000이면 1000000달러이고 약 10억원이다.

```
Connection Succeed
----- 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
2
----- TYPE 2 -----
** Show sales trends for various brands over the past k months. **
Which K? : 2
=====
brand_name    unit    dollar    month
=====
Bentley       1       26000     4
Audi          1       11000     4
Lamborghini   1       39000     4
Volkswagen    1       4900     4
Audi          2       30600     5
Lamborghini   1       26000     5
Volkswagen    2       9800     5
----- TYPE 2-1 -----
** Break this data out by gender of the buyer **
** 0 : skip **
** 1 : show results **
Input : 1
=====
brand_name    unit    dollar    month    gender
=====
Lamborghini   1       39000     4       M
Audi          1       11000     4       F
Audi          1       5000     5       F
Audi          1       25600     5       M
----- TYPE 2-1-1 -----
** Break this data out by gender of the buyer and then break out by income **
** 0 : skip **
** 1 : show results **
Input : 1
=====
brand_name    unit    dollar    month    gender    income
=====
Lamborghini   1       39000     4       M       9000
Audi          1       11000     4       F       7000
Audi          1       25600     5       M       8000
Audi          1       5000     5       F       7000
```

(TYPE 2)

타입 2를 선택하면 지난 몇 개월간의 트렌드를 볼 것인지 입력 받는다. 위의 예제에서는 전체 브랜드의 지난 2개월간 판매 트렌드를 보여준다. 6월을 기준으로 2개월인 4월, 5월의 판매 자동차 수(유닛)와 판매액(달러)

리)를 표로 보여준다. 월은 오름차순으로 정렬되어 출력된다. 쿼리에서 단순히 월만 보고 6에서 구입 날짜의 월을 빼는 방식을 사용하였다. 만약 현재 날짜에서 30일 단위의 지난 몇 개월간 기록을 보려고 한다면 TIMESTAMPDIFF() 함수를 사용하면 된다. WHERE절만 수정하면 된다. which k? 옆에 0을 입력하면 쿼리를 나갈 수 있다고 추가했다.

(TYPE 2-1)

타입 2의 결과가 끝나면 2-1을 볼 것인지 스킵할 것인지 입력해야 한다. 스킵을 원하면 0을, 결과를 보려면 1을 입력한다. 결과를 보기위해 0이 아닌 다른 숫자를 입력해도 되지만 숫자만 입력이 가능하다. 1을 입력하면 타입 1의 결과를 성별로 그룹화하여 결과를 보여준다. 4월, 5월의 각 판매 자동차 수와 판매액을 성별로 나누어 보여준다. 그룹화가 (연도 + 성별)로 되었다고 보면 된다. 월은 오름차순으로 정렬되어 출력된다.

(TYPE 2-1-1)

타입 2-1의 결과가 끝나면 2-1-1을 볼 것인지 스킵할 것인지 입력해야 한다. 스킵을 원하면 0을, 결과를 보려면 1을 입력한다. 결과를 보기위해 0이 아닌 다른 숫자를 입력해도 되지만 숫자만 입력이 가능하다. 1을 입력하면 타입 2-1의 결과를 연간 수입으로 그룹화하여 결과를 보여준다. 그룹화가 (연도 + 성별 + 연수입)로 되었다고 보면 된다. 수입과 판매액은 단위가 100달러이다. 예를 들어 10000이면 1000000달러이고 약 10억원이다.

```

3
----- TYPE 3 -----
** Find that transmissions made by supplier (company name) between two given dates are defective. **
** 0 : exit this query **
** 1 : continue this query **
Input : 1
Supplier name? : Btsup
Which start date? (yyyy-mm-dd): 2021-04-02
Which finish date? (yyyy-mm-dd): 2021-04-08
All information
=====
supplier_name    part_type        VIN              supply_date
=====
Btsup            transmission     KV1MB0000001    2021-04-02
Btsup            transmission     KV1MB0000004    2021-04-02
Btsup            transmission     KV1MB0000005    2021-04-04
Btsup            transmission     KV1MB0000006    2021-04-08
Record of suppliment from plants to suppliers
=====
plant_name       supplier_name    part_type        manufacture_date
=====
Btmp             Btsup            transmission     2018-02-02
Btmp             Btsup            transmission     2019-02-02
Btmp             Btsup            transmission     2020-02-02
Btmp             Btsup            transmission     2021-02-02

```

```

----- Subtypes in TYPE 3 -----
1. TYPE 3-1.
2. TYPE 3-2.
-----
Which subtype? (0 : exit this query) : 1
----- TYPE 3-1 -----
=====
VIN              customer_name    customer_phone    supply_date    sale_date
=====
KV1MB0000001    John             778899            2021-04-02     2018-05-09
KV1MB0000004    Blotem           998877            2021-04-02     2021-04-09
KV1MB0000005    Suite            949596            2021-04-04     2021-05-09
KV1MB0000006    Project          979699            2021-04-08     2021-05-09
----- Subtypes in TYPE 3 -----
1. TYPE 3-1.
2. TYPE 3-2.
-----
Which subtype? (0 : exit this query) : 2
----- TYPE 3-2 -----
=====
dealer           VIN              transmission     supply_date    sale_date
=====
MeisterMotors Seocho KV1MB0000001    DCT7            2021-04-02     2018-05-09
MeisterMotors Seocho KV1MB0000004    DCT7            2021-04-02     2021-04-09
MeisterMotors Seocho KV1MB0000005    DCT7            2021-04-04     2021-05-09
MeisterMotors Seocho KV1MB0000006    DCT7            2021-04-08     2021-05-09

```

(TYPE 3)

타입 3을 선택하면 쿼리를 계속 진행할 것인지 입력 받는다. 1을 입력하면 supplier 이름을 입력 받는다. 다음으로 기간을 입력 받는데 시작 날짜와 끝 날짜를 형식에 맞게 입력 받는다. 위의 예제에서는 Btsup이라는 변속기 공급점에서 결함이 있었고 해당 날짜는 2021년 4월 2일에서 8일까지라고 한다. 전체적인 정보를 우선 출력해준다. 공급점 이름과 부품 종류, 공급된 차량번호와 공급 날짜를 알 수 있다. 다음으로 출력된 표는 제조 공장이 공급점에 부품을 전달한 정보를 출력한다. 본 프로젝트에서는 중요한 부분은 아니지만 데이터 베이스 설계시 제조 공장에서 부품을 공급점에 공급할 수 있다고 가정하여서 해당 내용을 추가하였다. 제조 공장에서 부품을 만들어서 공급점으로 공급한 경우도 있으므로 사용자는 이 정보를 참고하여 결함이 생긴 부품이 공장에서 온 것인지 확인해볼 수 있다. 쿼리에서 WHERE절에 변속기에 해당하는 정보만 보도록 조건을 주었는데 엔진으로 변경하면 엔진에 결함이 생긴 경우도 다룰 수 있다.

(TYPE 3-1)

타입 3의 결과가 끝나면 3-1을 볼 것인지 3-2를 볼 것인지 서브 쿼리를 선택해야 한다. 0을 입력하면 쿼리를 빠져나갈 수 있다. 1을 입력하면 3-1이 실행된다. 결함이 있는 변속기가 장착된 차량의 번호와 그 차량을 구매한 고객의 이름과 전화번호가 출력이 된다. 위의 예제에서는 Btsup에서 공급한 변속기를 장착하고 있는 차량의 VIN, 고객이름, 고객 전화번호, 변속기 공급 날짜, 차량 판매 날짜를 보여준다. 차량은 이미 팔린 차량에 대해서만 쿼리를 적용한 것이다.

(TYPE 3-2)

2를 입력하면 3-2가 실행된다. 결함이 있는 변속기가 장착된 자동차를 판매한 딜러점의 정보를 출력하고 변속기의 종류를 출력하여 알려준다. 위의 예제에서는 결함이 있는 변속기가 장착된 차량을 판매한 딜러점의 이름과 VIN, 변속기 종류, 변속기 공급 날짜와 차량 판매 날짜를 보여준다. 모두 마이스터 모터스 서초점의 자동차들로 해당 VIN을 가진 자동차는 폭스바겐 티구안이다. 따라서 변속기 종류는 DCT7로 동일하다.

```
4
----- TYPE 4 -----
** Find the top k brands by dollar-amount sold by the year **
Which K? (0 : exit this query) : 3
Which year? : 2018
=====
brand_name      dollar
=====
Lamborghini     65000
Volkswagen      26400
Bentley         26000
Which K? (0 : exit this query) : 3
Which year? : 2019
=====
brand_name      dollar
=====
Lamborghini     65000
Audi            56800
Bentley         52000
Which K? (0 : exit this query) : 2
Which year? : 2020
=====
brand_name      dollar
=====
Lamborghini     39000
Audi            33800
```

(TYPE 4)

타입 4를 입력하면 top k에서 k를 입력받는다. 0을 입력하면 쿼리를 빠져나갈 수 있다. 다음으로 연도를 입

력받는다. 위의 예제에서는 2018년에 달러 판매액 기준 상위 3개 브랜드를 보여준다. 다음으로 또 쿼리가 반복되는데 차례로 2019년 상위 3개, 2020년 상위 2개 브랜드를 알 수 있다.

```

5
----- TYPE 5 -----
** Find the top k brands by until sales by the year **
Which K? (0 : exit this query) : 3
Which year? : 2018
=====
brand_name      unit
=====
Volkswagen      6
Audi             3
Lamborghini     2

Which K? (0 : exit this query) : 3
Which year? : 2019
=====
brand_name      unit
=====
Audi            5
Volkswagen      2
Lamborghini     2

Which K? (0 : exit this query) : 2
Which year? : 2020
=====
brand_name      unit
=====
Audi            4
Volkswagen      3

```

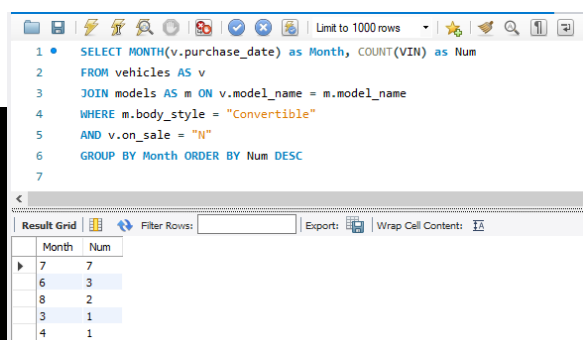
(TYPE 5)

타입 5를 입력하면 top k에서 k를 입력받는다. 0을 입력하면 쿼리를 빠져나갈 수 있다. 다음으로 연도를 입력받는다. 위의 예제에서는 2018년에 판매 자동차 수 기준 상위 3개 브랜드를 보여준다. 다음으로 또 쿼리가 반복되는데 차례로 2019년 상위 3개, 2020년 상위 2개 브랜드를 알 수 있다.

```

6
----- TYPE 6 -----
** In what month(s) do convertibles sell best? **
** 0 : exit this query **
** 1 : continue this query **
Input : 1
Best month is 7 (unit : 7)

```



The screenshot shows a SQL query in a text editor:

```

1 SELECT MONTH(v.purchase_date) as Month, COUNT(VIN) as Num
2 FROM vehicles AS v
3 JOIN models AS m ON v.model_name = m.model_name
4 WHERE m.body_style = "Convertible"
5 AND v.on_sale = "N"
6 GROUP BY Month ORDER BY Num DESC
7

```

Below the query is a 'Result Grid' table:

| Month | Num |
|-------|-----|
| 7     | 7   |
| 6     | 3   |
| 8     | 2   |
| 3     | 1   |
| 4     | 1   |

(TYPE 6)

타입 6을 입력하면 쿼리를 계속 진행할 것인지 입력 받는다. 1을 입력하면 몇 월에 컨버터블이 가장 많이 팔렸는지에 대한 결과가 출력된다. 위의 예제에서는 7월이 7대로 가장 많이 팔렸다. 만약 가장 많이 팔린 달이 2개 이상이면 모두 출력되도록 코드를 작성하였다. 실제로 7월이 맞는지 데이터베이스에서 확인해보았더니 올바른 결과였다.

```
7
----- TYPE 7 -----
** Find those dealers who keep a vehicle in inventory for the longest average time. **
** 0 : exit this query **
** 1 : continue this query **
Input : 1

Dealer is MeisterMotors Sinchon (day : 38.0000) , KlasseAuto Suwon (day : 38.0000)

1 • SELECT dealer_name, d_address, AVG(timestampdiff(DAY, production_date, CURDATE())) as avg_date
2 FROM vehicles
3 WHERE on_sale = "Y"
4 GROUP BY dealer_name, d_address
5 ORDER BY avg_date DESC
6

Result Grid | Filter Rows: | Export: | Wrap Cell Content:
dealer_name d_address avg_date
KlasseAuto Suwon 38.0000
MeisterMotors Sinchon 38.0000
Gojinmotors Suwon 35.0000
Teianmotors Ilisan 34.5000
```

(TYPE 7)

타입 7 입력하면 쿼리를 계속 진행할 것인지 입력 받는다. 1을 입력하면 평균적으로 가장 오랜 시간 inventory에 자동차들을 보관하고 있는 딜러점들을 출력한다. 위의 예제에서는 마이스터모터스 신촌점과 클라쎄오토 수원점이 평균 38일으로 가장 오랜 시간 보관중이었다. 만약 해당하는 딜러점이 2개 이상이면 모두 출력되도록 코드를 작성하였다. 실제로 결과가 맞는지 데이터베이스에서 확인해보았더니 올바른 결과였다.