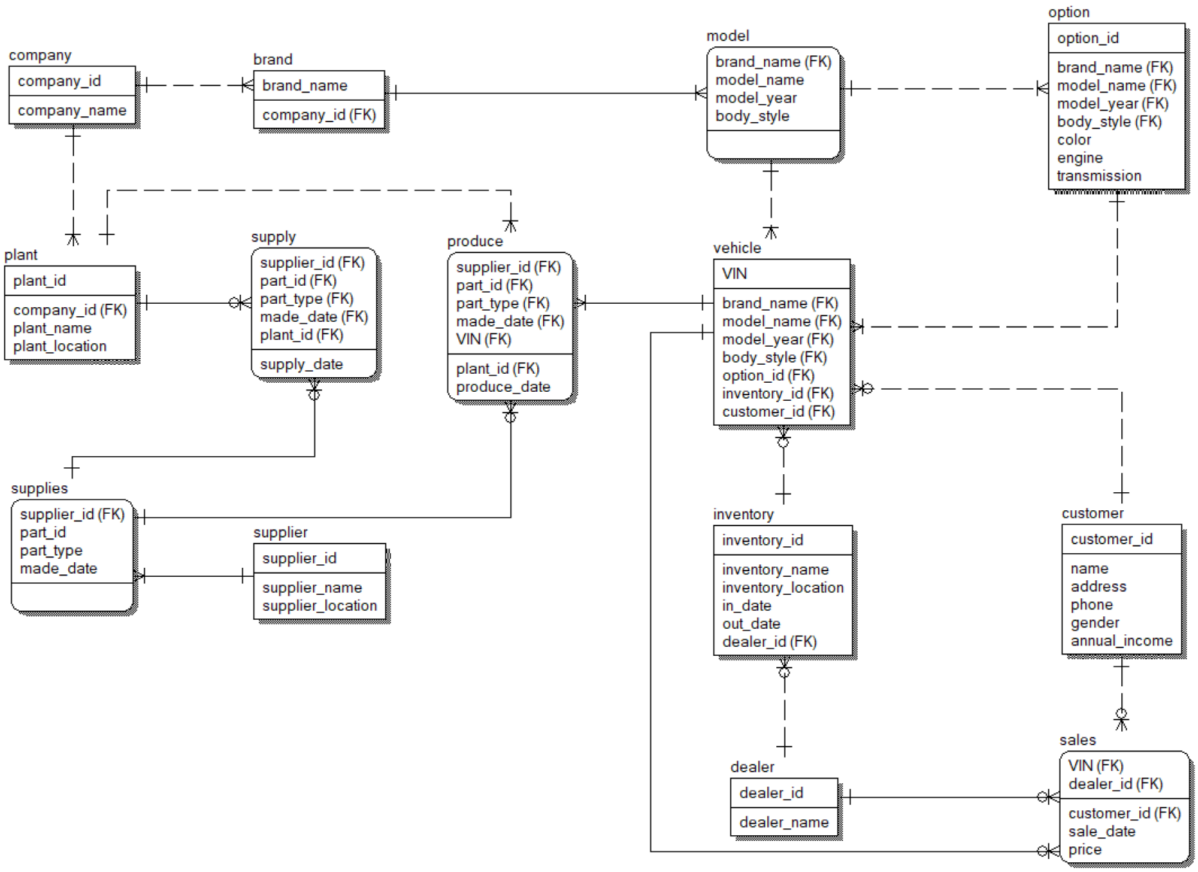
Database System Project 2
(Automobile company)

20160768 김홍엽

1. BCNF Decomposition



위 사진은 Project1에서 설계한 Relational Schema diagram이다. “vehicle”과 “sales”를 제외한 다른 모든 relation schema의 functional depencies는 모두 trivial 하거나, superkey이므로 BCNF이다. 하지만 vehicle과 sales relation schema는 BCNF 형태가 아니다. 따라서 이 두 relation을 BCNF로 분해하였다.

1.1. vehicle

vehicle의 F^+ 는

{ VIN -> brand_name, model_name, model_year, body_style, option_id, inventory_id, customer_id

option_id -> brand_name, model_name, model_year, body_style }이다.

VIN은 superkey이지만, option_id는 non-trivial 하면서 superkey도 아니다. 따라서 vehicle을 다음과 같이 분해하였다.

R1 = (option_id, brand_name, model_name, model_year, body_style)

R2 = (VIN, option_id, inventory_id, customer_id)

R1과 R2 모두 superkey만을 functional dependency로 갖는 BCNF 형태가 되었다.

하지만 R1의 경우, 이미 존재하는 option이라는 relation schema가 포함하고 있다. 따라서 R2만을 갖는 relation schema로 vehicle를 수정하였다.

1.2. sales

sales의 F^+ 는

{ VIN, dealer_id -> customer_id, sale_date, price

VIN -> customer_id}이다.

VIN, dealer_id는 superkey이지만, VIN은 non-trivial 하면서 superkey도 아니다. 따라서 sales를 다음과 같이 분해하였다.

R1 = (VIN, customer_id)

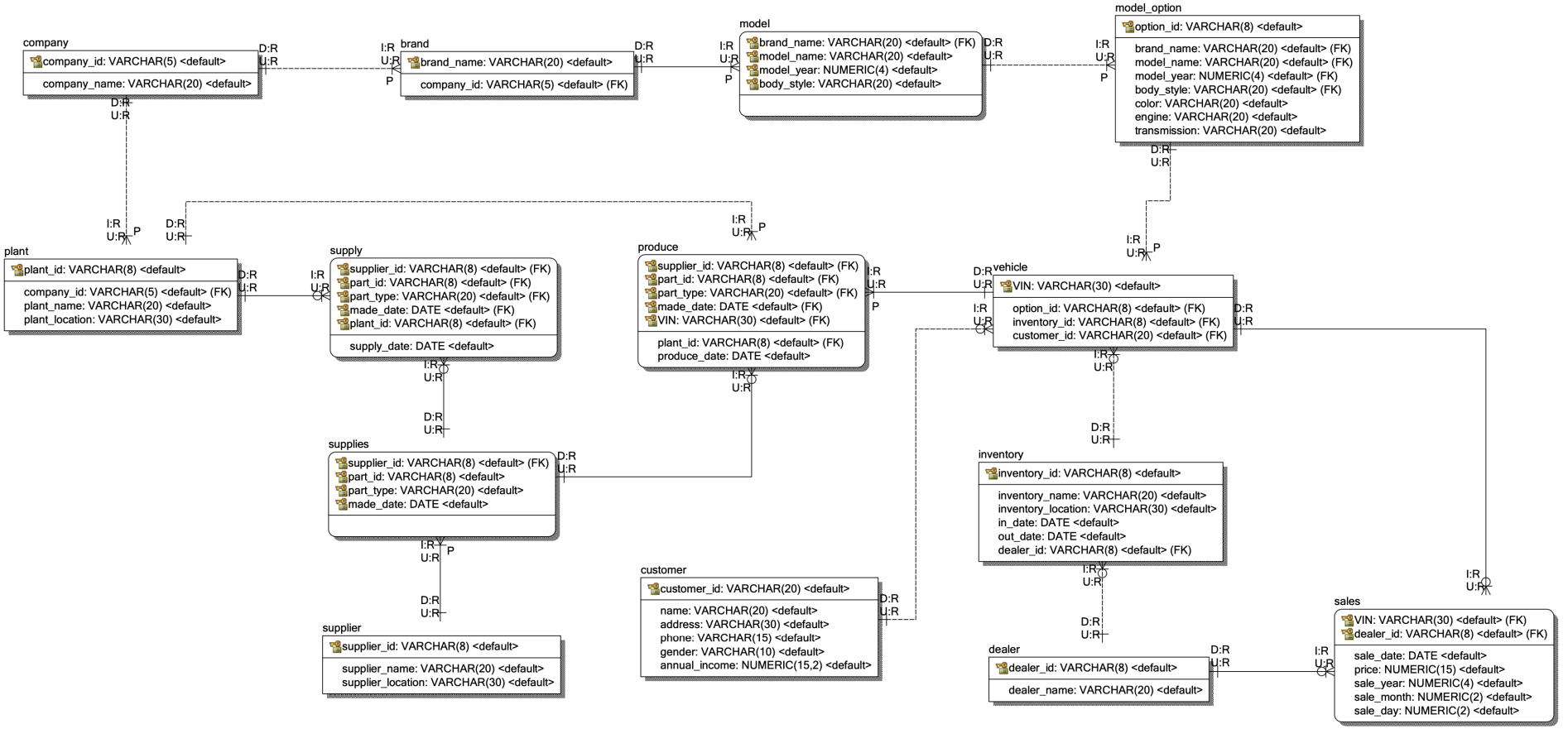
R2 = (VIN, dealer_id, sale_date, price)

R1과 R2 모두 superkey만을 functional dependency로 갖는 BCNF 형태가 되었다.

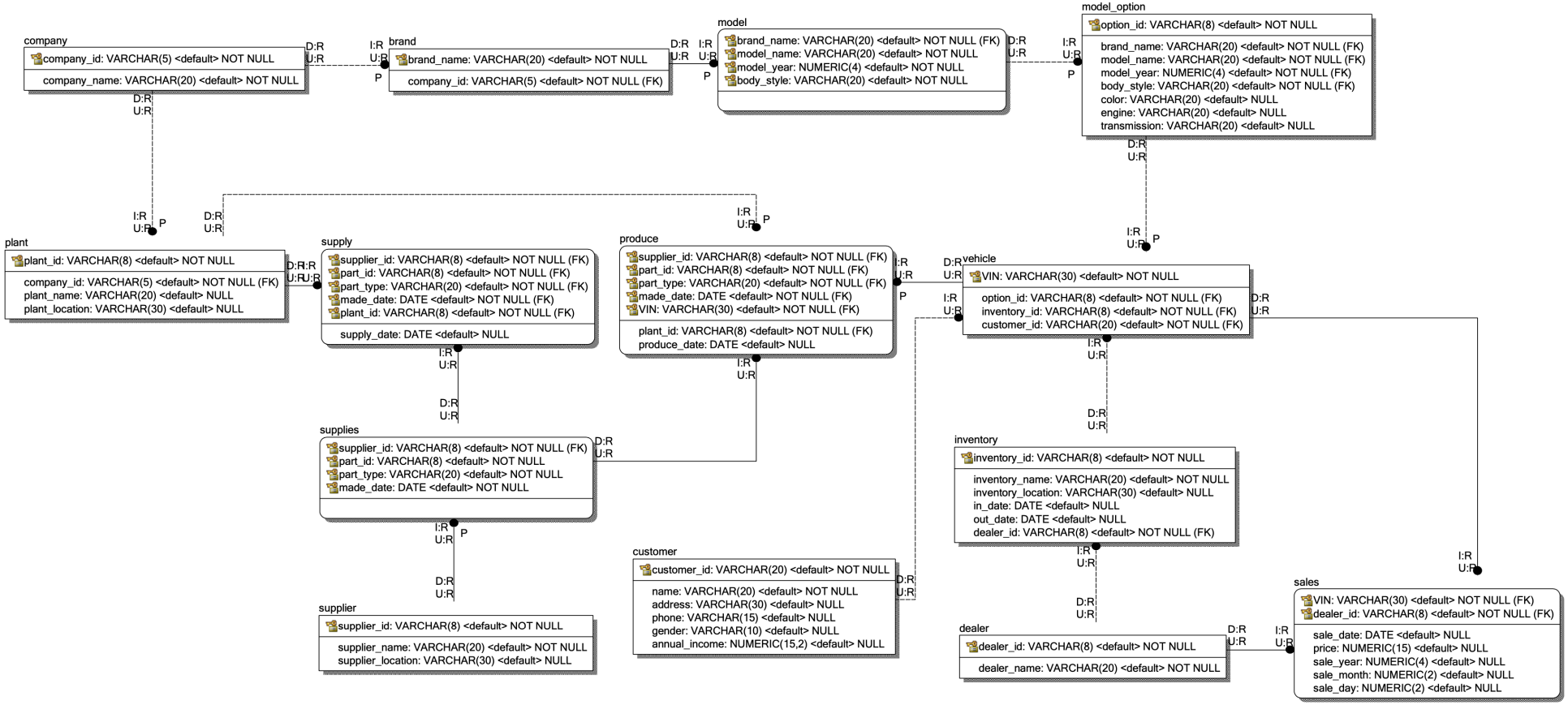
하지만 R1의 경우, 이미 존재하는 vehicle이라는 relation schema가 포함하고 있다. 따라서 R2만을 갖는 relation schema로 sales를 수정하였다.

최종 Decomposed Logical Schema diagram과 Physical Schema diagram은 다음과 같다.

Logical Schema diagram



Physical Schema diagram



vehicle과 sales가 수정되었고, “option”이 “model_option”으로 이름이 변경되었다. model_option으로 이름을 변경한 이유는 SQL query 작성 시 option 커맨드와의 혼동을 피하기 위해서이다. 추가로 sales에 sale_year, sale_month, sale_day 컬럼을 추가하였다.

위 Schema diagram을 통해 각 컬럼의 data types, domain(default), constraints(primary key, foreign key), relationship type(cardinality, identifying, non-identifying), allowing null 등을 확인할 수 있다. 각 테이블에 대한 자세한 설명은 Project1에서 다루었다.

2. ODBC implementation

위에서 설계한 Physical Schema를 ODBC 통해 modeling 하였고, 명세서의 쿼리문들을 실행할 수 있는 인터페이스를 코드로 구현하였다.

코드는 다음의 함수들로 구성되어 있다.

Function	Description
int initialize_DB()	input text 파일로부터 CRUD 쿼리들을 읽고 수행하여 DB를 초기화하는 함수이다. text 파일 맨 첫 줄은 table의 개수, 두 번째 줄은 삽입할 tuple의 총 개수를 나타낸다.
char *replaceAll(char *s, const char *olds, const char *news)	미리 입력된 쿼리문 문자열에 사용자의 입력 변수를 적용하기 위한 함수이다. 쿼리문의 특정 문자를 입력한 문자로 치환해준다.

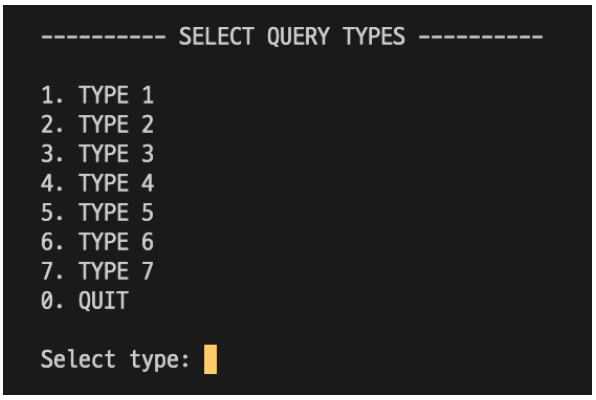
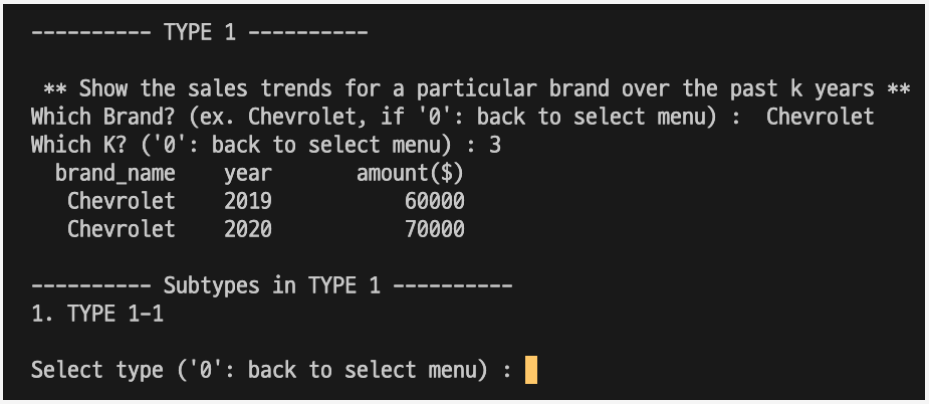
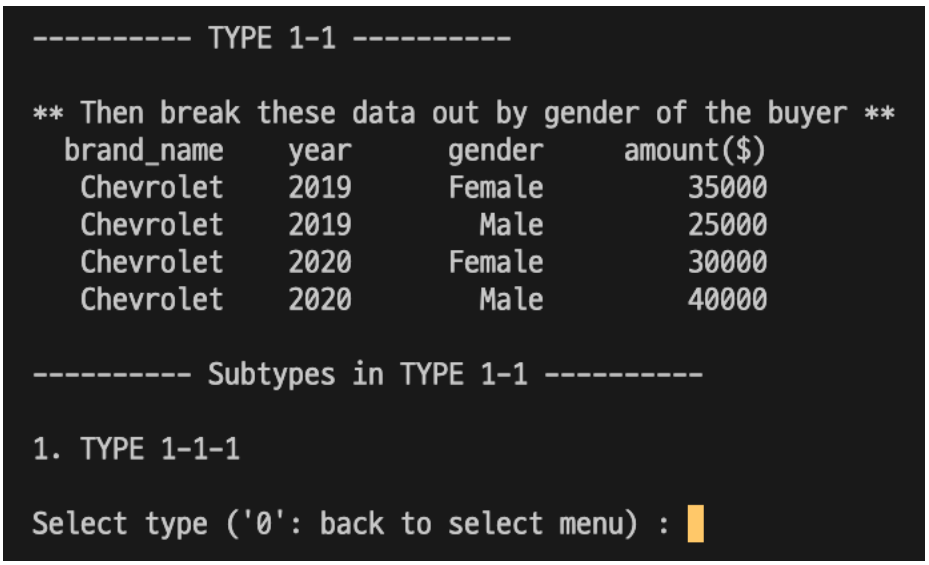
Function	Description
void type1(int k, char *brand_name, int type)	TYPE 1 쿼리문들을 실행하기 위한 함수이다. 매개변수 type에 따라 각각 다른 쿼리문이 실행된다. (TYPE 1: 1, TYPE 1-1: 2, TYPE 1-1-1: 3) TYPE 1의 쿼리문은 다음과 같다. *select brand_name, sale_year, sum(price)* *from (vehicle join model_option using (option_id)) join sales using (VIN)* *where brand_name = 'BRAND' and year(sale_date) >= year(date_sub(now(), interval K year)) and year(sale_date) < year(now())* *group by brand_name, sale_year* *order by sale_year* vehicle과 model_option, sales 테이블을 join하여 결과를 도출한다. BRAND와 K에는 사용자의 입력값이 들어가게 된다. TYPE 1-1의 쿼리문은 다음과 같다. select brand_name, sale_year, gender, sum(price) from ((vehicle join model_option using (option_id)) join sales using (VIN)) join customer using (customer_id) where brand_name = 'BRAND' and year(sale_date) >= year(date_sub(now(), interval K year)) and year(sale_date) < year(now()) group by brand_name, sale_year, gender order by sale_year, gender TYPE 1의 쿼리문에서 추가로 customer 테이블과 join하여, 구매자의 성별에 따라 sales trend가 추가로 나뉘어지도록 하였다. TYPE 1-1-1의 쿼리문은 다음과 같다. *select brand_name, sale_year, gender,* *case* *when annual_income between 0 and 19999 then 1* *when annual_income between 20000 and 49999 then 2* *when annual_income between 50000 and 79999 then 3* *else 4* *end income_range,* *sum(price)* *from ((vehicle join model_option using (option_id)) join sales using (VIN)) join customer using (customer_id)* *where brand_name = 'BRAND' and year(sale_date) >= year(date_sub(now(), interval K year)) and year(sale_date) < year(now())* *group by brand_name, sale_year, gender, income_range* *order by sale_year, gender, income_range* TYPE 1-1의 쿼리문에서 추가로 case문을 통해 구매자의 annual_income의 범위에 따라 income_range를 설정한 후, 그에 따라 sales trend가 추가로 나뉘어지도록 하였다. income_range의 범위 기준은 다음과 같다. income_range 1: 0 <= income < 20000($) income_range 2: 20000($) <= income < 50000($) income_range 3: 50000($) <= income < 80000($) income_range 4: 80000($) <= income

Function	Description
void type2(int k, int type)	TYPE 2 쿼리문들을 실행하기 위한 함수이다. 매개변수 type에 따라 각각 다른 쿼리문이 실행된다. (TYPE 2: 1, TYPE 2-1: 2, TYPE 2-1-1: 3) TYPE 2의 쿼리문은 다음과 같다. *select brand_name, sale_year, sale_month, sum(price)* *from (vehicle join model_option using (option_id)) join sales using (VIN)* *where sale_date > last_day(date_sub(date_sub(now(), interval K month), interval 1 month)) and sale_date <= last_day(date_sub(now(), interval 1 month))* *group by brand_name, sale_year, sale_month* *order by brand_name, sale_year, sale_month* vehicle과 model_option, sales를 join하여 결과를 도출한다. sale_date 컬럼을 통해 past k months에 대한 조건을 만들었다. K에는 사용자의 입력값이 들어가게 된다. TYPE 2-1의 쿼리문은 다음과 같다. *select brand_name, sale_year, sale_month, gender, sum(price)* *from ((vehicle join model_option using (option_id)) join sales using (VIN)) join customer using (customer_id)* *where sale_date > last_day(date_sub(date_sub(now(), interval K month), interval 1 month)) and sale_date <= last_day(date_sub(now(), interval 1 month))* *group by brand_name, sale_year, sale_month, gender* *order by brand_name, sale_year, sale_month, gender* TYPE 2의 쿼리문에서 추가로 customer 테이블과 join하여, 구매자의 성별에 따라 sales trend가 추가로 나뉘어지도록 하였다. TYPE 2-1-1의 쿼리문은 다음과 같다. *select brand_name, sale_year, sale_month, gender,* *case* *when annual_income between 0 and 19999 then 1* *when annual_income between 20000 and 49999 then 2* *when annual_income between 50000 and 79999 then 3* *else 4* *end income_range,* *sum(price)* from ((vehicle join model_option using (option_id)) join sales using (VIN)) *join customer using (customer_id)* *where sale_date > last_day(date_sub(date_sub(now(), interval K month), interval 1 month)) and sale_date <= last_day(date_sub(now(), interval 1 month))* *group by brand_name, sale_year, sale_month, gender, income_range* *order by brand_name, sale_year, sale_month, gender, income_range* TYPE 2-1의 쿼리문에서 추가로 case문을 통해 구매자의 annual_income의 범위에 따라 income_range를 설정한 후, 그에 따라 sales trend가 추가로 나뉘어지도록 하였다. income_range의 범위 기준은 다음과 같다. income_range 1: 0 <= income < 20000($) income_range 2: 20000($) <= income < 50000($) income_range 3: 50000($) <= income < 80000($) income_range 4: 80000($) <= income

Function	Description
	TYPE 3 쿼리문들을 실행하기 위한 함수이다. 매개변수 type에 따라 각각 다른 쿼리문이 실행된다. (TYPE 3: 1, TYPE 3-1: 2, TYPE 3-2: 3) TYPE 3의 쿼리문은 다음과 같다. *select supplier_name, part_id, part_type, made_date* *from supplies join supplier using (supplier_id)* *where supplier_name = '**SUPPLIER**' and part_type = '**Transmission**' and made_date between '**DATE1**' and '**DATE2**'* *order by made_date* supplies와 supplier를 join하여 결과를 도출한다. SUPPLIER와 DATE1, DATE2는 사용자에게 입력받은 값으로 바뀌게 된다. TYPE 3-1의 쿼리문은 다음과 같다. *select supplier_name, part_id, part_type, made_date, VIN, customer_id, name as customer_name* *from ((produce join supplier using (supplier_id)) join vehicle using (VIN)) join customer using (customer_id)* *where supplier_name = '**SUPPLIER**' and part_type = '**Transmission**' and made_date between '**DATE1**' and '**DATE2**'* *order by made_date, VIN* produce, supplier, vehicle, customer 테이블들을 join하여 해당 transmission을 가지고 있는 차량의 VIN과 customer 정보를 도출한다. TYPE 3-2의 쿼리문은 다음과 같다. *select supplier_name, part_id, part_type, made_date, VIN, dealer_id, dealer_name* *from (((produce join supplier using (supplier_id)) join vehicle using (VIN)) join sales using (VIN)) join dealer using (dealer_id)* *where supplier_name = '**SUPPLIER**' and part_type = '**Transmission**' and made_date between '**DATE1**' and '**DATE2**'* *order by made_date, VIN, dealer_id* produce, supplier, vehicle, sales, dealer 테이블들을 join하여 해당 transmission을 가지고 있는 차량을 판매한 dealer 정보를 도출한다.
void type4(int k, int year)	TYPE 4 쿼리문을 실행하기 위한 함수이다. TYPE 4의 쿼리문은 다음과 같다. *select sale_year, brand_name, sum(price) as dollar_amount* *from (vehicle join sales using (VIN)) join model_option using (option_id)* *where sale_year = '**YEAR**'* *group by brand_name* *order by dollar_amount desc limit K* sales와 vehicle, model_option을 join하여 해당 년도에 판매한 가격의 합이 가장 많은 top K의 브랜드를 추출한다. YEAR과 K에는 사용자의 입력값이 들어간다.
void type5(int k, int year)	TYPE 5 쿼리문을 실행하기 위한 함수이다. TYPE 5의 쿼리문은 다음과 같다. *select sale_year, brand_name, count(distinct VIN) as unit_sales* *from (sales join vehicle using (VIN)) join model_option using (option_id)* *where sale_year = '2020'* *group by brand_name* *order by unit_sales desc limit K* sales와 vehicle, model_option을 join하여 해당 년도에 차량을 가장 많이 판매한 top K의 브랜드를 추출한다. YEAR과 K에는 사용자의 입력값이 들어간다.

Function	Description
void type6()	TYPE 6 쿼리문을 실행하기 위한 함수이다. TYPE 6의 쿼리문은 다음과 같다. *with t as (select sale_month, count(distinct VIN) as c_sale* *from (sales join vehicle using (VIN)) join model_option using (option_id)* *where body_style = 'Convertible' group by sale_month order by count(distinct VIN) desc limit 1)* *select sale_month, body_style, count(distinct VIN)* *from (sales join vehicle using (VIN)) join model_option using (option_id)* *where body_style = 'Convertible'* *group by sale_month, body_style* *having count(distinct VIN) >= (select c_sale from t)* *order by sale_month* 먼저 sales와 vehicle, model_option을 join하여 Convertible 차량의 판매 대수를 sale_month로 묶은 후 상위 1개의 값만을 갖고온다. 그 테이블을 임시 테이블 t로 정의하고, 똑같은 조건 하에 그 t의 차량 판매 대수보다 크거나 같은 값을 가진 sale_month를 추출하여 결과를 도출한다. 이렇게 복잡한 쿼리문을 구성한 이유는 단순히 차량 판매 대수로 정렬하여 “limit 1”로 추출하게 되면, 상위 1등과 같은 값을 가진 다른 sale_month의 정보를 추출할 수 없기 때문이다.
void type7()	TYPE 7 쿼리문을 실행하기 위한 함수이다. TYPE 7의 쿼리문은 다음과 같다. *with t as (select dealer_id, avg(datediff(out_date, in_date)) as avg_time* *from inventory* *group by dealer_id* *order by avg_time desc limit 1)* *select dealer_id, dealer_name, avg(datediff(out_date, in_date))* *from inventory join dealer using (dealer_id)* *group by dealer_id* *having avg(datediff(out_date, in_date)) >= (select avg_time from t)* 먼저 inventory table에서 dealer_id로 group by한 out_date, in_date 차이의 평균 값의 상위 1명의 값을 가져온다. 그 테이블을 임시 테이블 t로 정의하고, 똑같은 조건 하에 그 t의 시간보다 크거나 같은 값을 가진 dealer_id를 추출하여 결과를 도출한다. 이렇게 복잡한 쿼리문을 구성한 이유는 단순히 out_date, in_date 차이의 평균값으로 정렬하여 “limit 1” 로 추출하게 되면, 상위 1등과 같은 값을 가진 다른 dealer의 정보를 추출할 수 없기 때문이다.
void clrscr()	show_interface() 함수에서 콘솔의 내용을 지우기 위해 사용된다.
void show_interface()	사용자 인터페이스를 구성하는 함수이다. switch문을 통해 각 TYPE의 쿼리문 함수들을 실행하며, subtype의 쿼리문 또한 선택하여 실행할 수 있다. 사용자로부터 각 변수에 대한 정보를 입력받고, 각 쿼리문의 결과값을 콘솔에 출력해준다. 사용자로부터 ‘0’을 입력받으면, select menu로 돌아가게 되고, select menu 창에서 ‘0’을 입력하면 show_interface() 함수가 종료된다.
int drop_tables()	생성한 테이블의 tuple들을 모두 지우고, 테이블을 drop하는 함수이다.
int main(void)	프로그램을 실행하는 메인함수이다. 먼저 mysql과 연결을 하고, 성공적으로 연결이 되면, 입력 text 파일로부터 데이터베이스를 초기화한다. 그 뒤 show_interface() 함수가 실행되고, show_interface() 함수가 종료되면, drop_tables() 함수가 실행된다. 마지막으로 text 파일을 닫고, mysql과 연결을 끊은 뒤 프로그램을 종료한다.

3. Execution examples

Type	Examples
Select menu	
TYPE 1	
	

Type	Examples
	---------- TYPE 1-1-1 ---------- ** Then by income range ** income_range 1: 0 <= income < 20000($) income_range 2: 20000($) <= income < 50000($) income_range 3: 50000($) <= income < 80000($) income_range 4: 80000($) <= income brand_name year gender income_range amount($) Chevrolet 2019 Female 4 35000 Chevrolet 2019 Male 3 25000 Chevrolet 2020 Female 2 30000 Chevrolet 2020 Male 1 40000 Put '0' to go back to select menu:
TYPE 2	---------- TYPE 2 ---------- ** Show sales trends for various brands over the past k months ** Which K? ('0': back to select menu) : 20 brand_name year month amount($) Brand-1 2020 10 50000 Chevrolet 2020 7 40000 Chevrolet 2020 8 30000 Chevrolet 2021 4 20000 ---------- Subtypes in TYPE 2 ---------- 1. TYPE 2-1 Select type ('0': back to select menu) :
	---------- TYPE 2-1 ---------- ** Then break these data out by gender of the buyer ** brand_name year month gender amount($) Brand-1 2020 10 Female 15000 Brand-1 2020 10 Male 35000 Chevrolet 2020 7 Male 40000 Chevrolet 2020 8 Female 30000 Chevrolet 2021 4 Female 20000 ---------- Subtypes in TYPE 2-1 ---------- 1. TYPE 2-1-1 Select type ('0': back to select menu) :

Type	Examples
	---------- TYPE 2-1-1 ---------- ** Then by income range ** income_range 1: 0 <= income < 20000($) income_range 2: 20000($) <= income < 50000($) income_range 3: 50000($) <= income < 80000($) income_range 4: 80000($) <= income brand_name year month gender income_range amount($) Brand-1 2020 10 Female 4 15000 Brand-1 2020 10 Male 3 35000 Chevrolet 2020 7 Male 1 40000 Chevrolet 2020 8 Female 2 30000 Chevrolet 2021 4 Female 2 20000 Put '0' to go back to select menu:
TYPE 3	---------- TYPE 3 ---------- ** Find that transmissions made by supplier (company name) between two given dates are defective ** Which first date(YYYY-MM-DD)? ('0': back to select menu) : 2019-04-01 Which second date(YYYY-MM-DD)? ('0': back to select menu) : 2020-04-01 Which supplier? (ex. GM Supplier if '0': back to select menu) : GM Supplier Supplier (company name): GM Supplier Transmissions made between 2019-04-01 and 2020-04-01 supplier_name part_id part_type made_date GM Supplier ABCD0002 Transmission 2019-04-01 GM Supplier ABCD0003 Transmission 2019-05-01 GM Supplier ABCD0004 Transmission 2020-03-01 ---------- Subtypes in TYPE 3 ---------- 1. TYPE 3-1 2. TYPE 3-2 Select type ('0': back to select menu) :
	---------- TYPE 3-1 ---------- ** Find the VIN of each car containing such a transmission and the customer to which it was sold ** Supplier (company name): GM Supplier Transmissions made between 2019-04-01 and 2020-04-01 supplier_name part_id part_type made_date VIN customer_id customer_name GM Supplier ABCD0002 Transmission 2019-04-01 VIN0000002 CSE0002 Kim3 GM Supplier ABCD0003 Transmission 2019-05-01 VIN0000003 CSE0003 Kim4 GM Supplier ABCD0004 Transmission 2020-03-01 VIN0000004 CSE0004 Kim5 Put '0' to go back to select menu:
	---------- TYPE 3-2 ---------- ** Find the dealer who sold the VIN and transmission for each vehicle containing these transmissions ** Supplier (company name): GM Supplier Transmissions made between 2019-04-01 and 2020-04-01 supplier_name part_id part_type made_date VIN dealer_id dealer_name GM Supplier ABCD0002 Transmission 2019-04-01 VIN0000002 00000000 Hongyeob Kim1 GM Supplier ABCD0003 Transmission 2019-05-01 VIN0000003 00000000 Hongyeob Kim1 GM Supplier ABCD0004 Transmission 2020-03-01 VIN0000004 00000001 Jinwoo Kim1 Put '0' to go back to select menu:
TYPE 4	---------- TYPE 4 ---------- ** Find the top k brands by dollar-amount sold by the year ** Which year? ('0': back to select menu) : 2020 Which K? ('0': back to select menu) : 2

Type	Examples
	---------- TYPE 4 ---------- ** Find the top k brands by dollar-amount sold by the year ** Which year? ('0': back to select menu) : 2020 Which K? ('0': back to select menu) : 2
TYPE 5	---------- TYPE 5 ---------- ** Find the top k brands by unit sales by the year ** Which year? ('0': back to select menu) : 2019 Which K? ('0': back to select menu) : 1
	---------- TYPE 5 ---------- ** Find the top k brands by unit sales by the year ** TOP 1 brands (2019) sale_year brand_name unit_sales 2019 Brand-1 2 Put '0' to go back to select menu:
TYPE 6	---------- TYPE 6 ---------- ** In what month(s) do convertibles sell best? ** TOP month(s) convertibles sell best month body_style unit_sales 10 Convertible 2 Put '0' to go back to select menu:
TYPE 7	---------- TYPE 7 ---------- ** Find those dealers who keep a vehicle in inventory for the longest average time ** dealer_id dealer_name avg_time(day) 00000000 Hongyeob Kim1 393.2500 Put '0' to go back to select menu: