

Database system project2 report

# **Automobile Company Database design**

컴퓨터공학과 20171682 임정호

## 1. 본 프로젝트의 목표

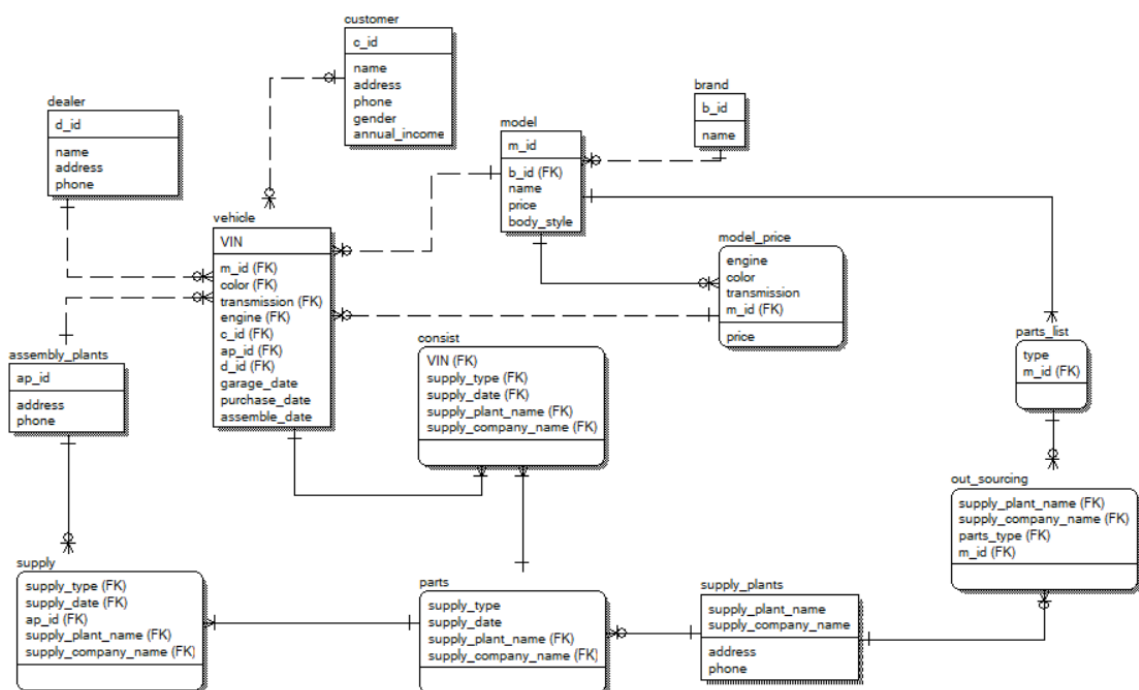
1. 프로젝트 1에서 설계한 logical schema diagram을 BCNF를 만족하게끔, relation을 decompose한다.
2. physical schema diagram을 만들어 각 relation에 맞는 구체적인 data type과 constraint, domain을 설정한다.
3. mysql과 visual studio 19를 ODBC를 이용하여 mysql 코드와 c코드과 적절하게 실행될 수 있게 한다.
4. 명세서에 주어진 13개의 문제를 해결할 수 있는 query를 작성하여 각 query에 맞는 결과를 출력할 수 있다.

## 2. BCNF 정규화

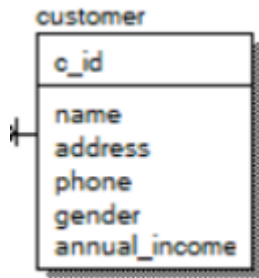
프로젝트 1에서 설계했던 logical schema diagram에서 일부 relation의 수정이 있었다.

1개의 relation이 삭제되었고, 1개의 relation이 decompose되었다.

수정된 결과는 다음과 같다.



## 2.1 customer



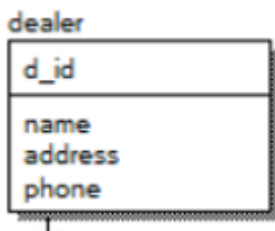
customer relation은 아래와 같이 나타낼 수 있다.

$R1 = \text{customer}(c\_id, \text{name}, \text{address}, \text{phone}, \text{gender}, \text{annual\_income})$

$F1 = \{ c\_id \rightarrow \text{name}, \text{address}, \text{phone}, \text{gender}, \text{annual\_income} \}$

위 relation에서 functional dependency는 c\_id가 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 c\_id는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.2 dealer



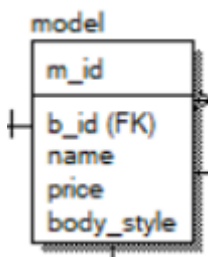
dealer의 relation은 아래와 같이 나타낼 수 있다.

$R2 = \text{dealer}(d\_id, \text{name}, \text{address}, \text{phone})$

$F2 = \{ d\_id \rightarrow \text{name}, \text{address}, \text{phone} \}$

위 relation에서 functional dependency는 d\_id가 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 d\_id는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.3 model



model의 relation은 아래와 같이 나타낼 수 있다.

$R3 = \text{model}(m\_id, \text{name}, \text{price}, b\_id, \text{body\_style})$

$F3 = \{ m\_id \rightarrow \text{name}, \text{price}, b\_id, \text{body\_style} \}$

위 relation에서 functional dependency는 m\_id가 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 m\_id는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.4 vehicle

vehicle의 relation은 초기 아래와 같은 형태의 relation이었다.

R4 = vehicle( VIN, m\_id, c\_id, d\_id, ap\_id, garage\_date, purchase\_date, assemble\_date, engine, color, transmission, price)

F4 = { VIN -> m\_id, c\_id, d\_id, ap\_id, garage\_date, purchase\_date, assemble\_date, engine, color, transmission, price }

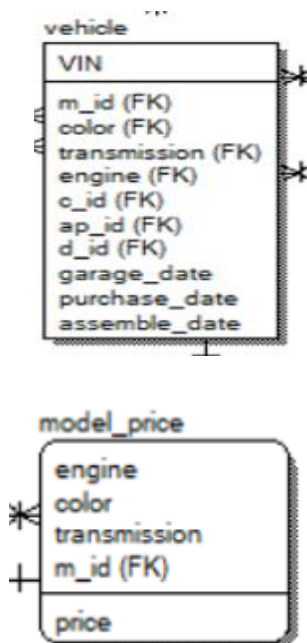
F4+ = { VIN -> m\_id, c\_id, d\_id, ap\_id, garage\_date, purchase\_date, assemble\_date, engine, color, transmission, price

VIN, d\_id -> garage\_date,

VIN, ap\_id -> assemble\_date, engine, color, transmission,

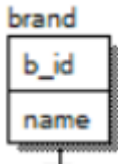
m\_id, engine, color, transmission -> price }

위에서 순서대로 1번부터 4번으로 두면, 1번 functional dependency는 VIN이 super key이므로 BCNF를 위배하지 않는다. 2번 functional dependency는 VIN과 d\_id ( 딜러 id )를 알면 차고에 입고된 날짜를 알 수 있다는 fd인데, VIN, d\_id는 superkey이므로 BCNF를 위배하지 않는다. 3번 fd는 VIN과 ap\_id ( assmble plant id )를 알면 차가 조립된 날짜와 그 차의 option ( 엔진, 색상, 변속기 )를 알 수 있다는 fd이다. 이 또한 VIN과 ap\_id가 super key이므로 BCNF를 위배하지 않는다. 4번 fd는 model id와 차의 option을 알면 그 차의 가격을 결정할 수 있다는 의미의 fd이다. 이 때 m\_id, engine, color, transmission은 super key도 아니고, trivial한 dependency도 아니다. 따라서 4번 fd는 BCNF를 위배한다. 따라서 decomposition이 필요하다.



그래서 model\_price = ( m\_id, engine, color, transmission, price )과, vehicle = ( VIN, m\_id, c\_id, d\_id, ap\_id, garage\_date, purchase\_date, assemble\_date, engine, color, transmission)으로 2개의 relation으로 decompose하였다. 이 때 vehicle은 BCNF를 위배하지 않고, decompose된 relation, model\_price 또한 dependency가 m\_id, engine, color, transmission -> price밖에 존재하지 않고, m\_id, engine, color, transmission은 super key이므로 BCNF를 위배하지 않는다. vehicle ∩ model\_price -> model\_price 이므로 lossless한 decomposition임을 확인할 수 있다. 2개의 relation이다. 추가로 프로젝트 1에서 option이라는 relation이 있었는데, model\_price와 중복되어 삭제하였다.

## 2.5 brand



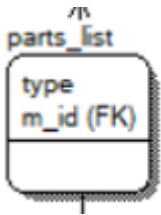
brand 의 relation은 아래와 같이 나타낼 수 있다.

$R5 = \text{brand}(b\_id, \text{name})$

$F5 = \{ b\_id \rightarrow \text{name} \}$

위 relation에서 functional dependency는 b\_id가 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 b\_id는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.6 parts\_list



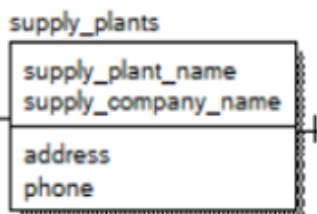
parts\_list 의 relation은 아래와 같이 나타낼 수 있다.

$R6 = \text{parts\_list}(m\_id, \text{type})$

$F6 = \{ m\_id, \text{type} \rightarrow m\_id, \text{type} \}$

위 relation에서 functional dependency는 (m\_id, type) -> (m\_id, type)인 dependency밖에 없다. 이 때 dependency는 trivial한 dependency 이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.7 supply\_plants



supply\_plants의 relation은 아래와 같이 나타낼 수 있다.

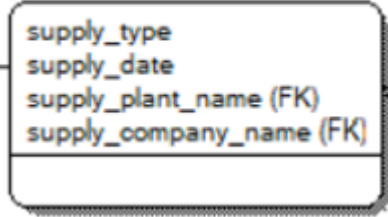
$R7 = \text{supply\_plants}(\text{supply\_plant\_name}, \text{supply\_company\_name}, \text{address}, \text{phone})$

$F7 = \{ \text{supply\_plant\_name}, \text{supply\_company\_name} \rightarrow \text{address}, \text{phone} \}$

위 relation에서 functional dependency는 (supply\_plant\_name, supply\_company\_name)이 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 supply\_plant\_name, supply\_company\_name는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.8 parts

parts



parts의 relation은 아래와 같이 나타낼 수 있다.

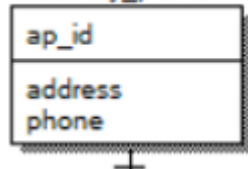
$R8 = \text{parts}(\text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date})$

$F8 = \{ \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date} \rightarrow \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date} \}$

위 relation에서 functional dependency는 자기 자신이 자기 자신을 결정하는 dependency밖에 없다. 이 때 trivial한 dependency이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.9 assembly\_plants

assembly\_plants



assembly\_plants의 relation은 아래와 같이 나타낼 수 있다.

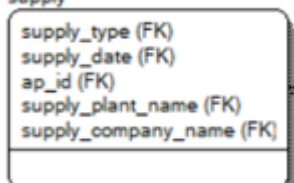
$R9 = \text{assembly\_plants}(\text{a\_id}, \text{address}, \text{phone})$

$F9 = \{ \text{ap\_id} \rightarrow \text{address}, \text{phone} \}$

위 relation에서 functional dependency는 ap\_id가 나머지 attribute를 모두 결정하는 dependency밖에 없다. 이 때 ap\_id는 super key이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.10 supply

supply



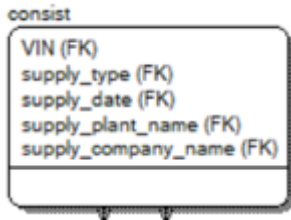
supply의 relation은 아래와 같이 나타낼 수 있다.

$R10 = \text{supply}(\text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date}, \text{ap\_id})$

$F8 = \{ \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date}, \text{ap\_id} \rightarrow \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date}, \text{ap\_id}, \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date} \rightarrow \text{supply\_plant\_name}, \text{supply\_company\_name}, \text{supply\_type}, \text{supply\_date}} \}$

위 relation에서 functional dependency는 자기 자신이 자기 자신을 결정하는 dependency들 밖에 없다. 이 때 모두 trivial한 dependency이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.11 consist



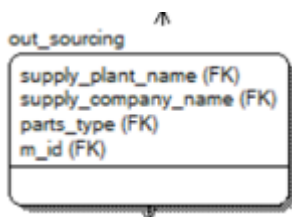
consist의 relation은 아래와 같이 나타낼 수 있다.

R11 = consist( VIN, supply\_plant\_name, supply\_company\_name, supply\_type, supply\_date )

F11 = { supply\_plant\_name, supply\_company\_name, supply\_type, supply\_date, VIN -> supply\_plant\_name, supply\_company\_name, supply\_type, supply\_date, VIN,  
supply\_plant\_name, supply\_company\_name, supply\_type, supply\_date -> supply\_plant\_name, supply\_company\_name, supply\_type, supply\_date }

위 relation에서 functional dependency는 자기 자신이 자기 자신을 결정하는 dependency밖에 없다. 이 때 모두 trivial한 dependency이므로 BCNF를 만족하는 것을 확인할 수 있다.

## 2.12 out\_sourcing



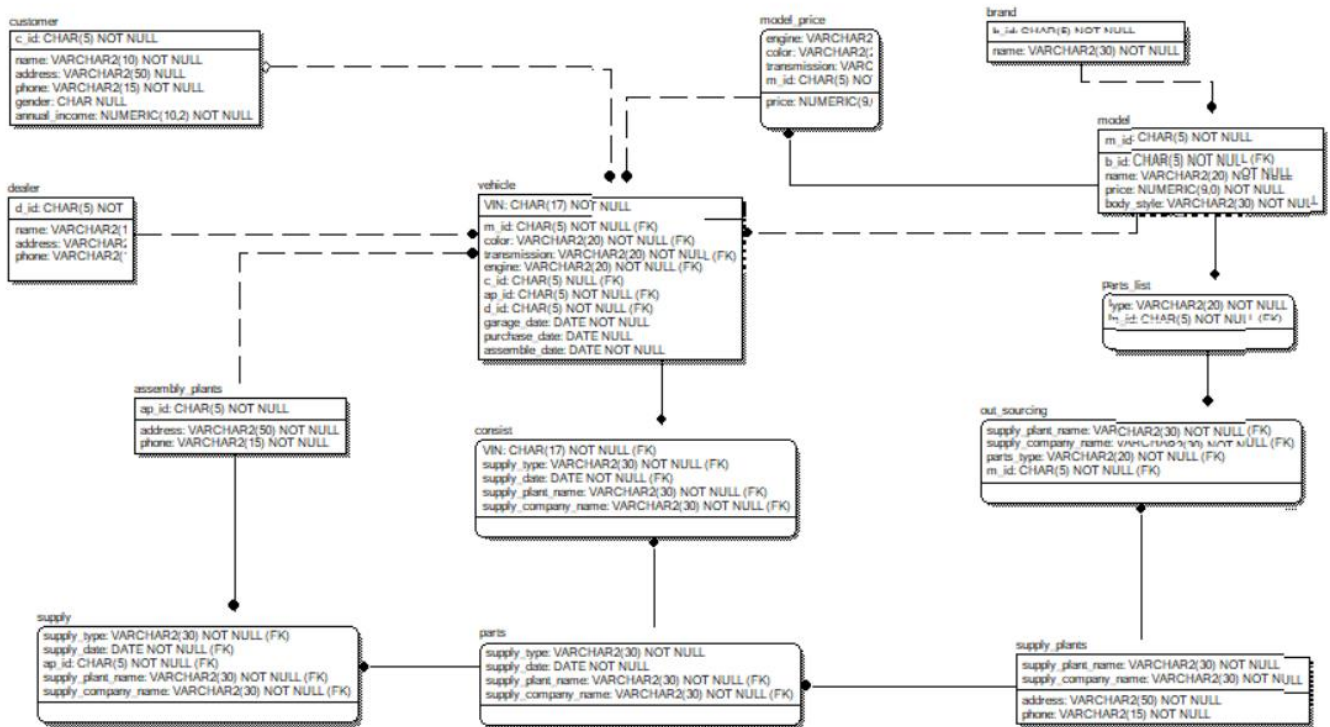
out\_sourcing의 relation은 아래와 같이 나타낼 수 있다.

R12 = out\_sourcing( m\_id, supply\_plant\_name, supply\_company\_name, parts\_type )

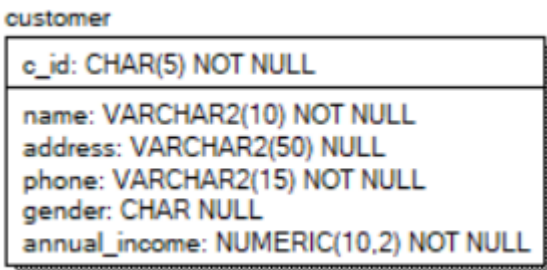
F12 = { supply\_plant\_name, supply\_company\_name, parts\_type, supply\_date, m\_id -> supply\_plant\_name, supply\_company\_name, parts\_type, supply\_date, m\_id,  
m\_id, parts\_type -> m\_id, parts\_type }

위 relation에서 functional dependency는 자기 자신이 자기 자신을 결정하는 dependency밖에 없다. 이 때 모두 trivial한 dependency이므로 BCNF를 만족하는 것을 확인할 수 있다.

### 3. Physical schema diagram



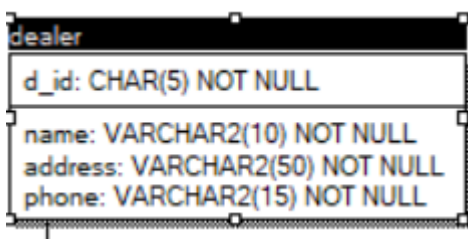
#### 3.1 customer



c\_id는 5자리의 id값을 가지는데, c\_id의 경우는 2XXXX의 값을 가질 수 있도록 constraint를 설정하였다.

name, phone은 null이 들어오지 못 하도록 하였고, address는 null 값을 허용하였다. gender는 char로 두었는데, 고객이 회사인 경우를 고려해서 null을 허용했다. annual\_income은 양수인 값만 들어올 수 있게 constraint를 설정하였다.

#### 3.2 dealer

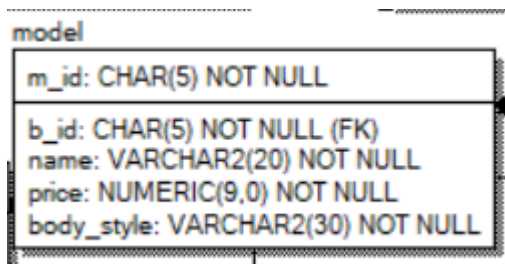


d\_id는 char(5)로 5자리의 id값을 가지는데, 6XXXX의 값을 가질 수 있도록 constraint를 설정하였다.

name, address, phone은 null이 들어오지 못 하도록 하였고, address를null 값을 허용하지 않은 이유는 차가 조립되면 차고로 보내야되는데 그 때 dealer 차고의 주소가 필요하기 때문이다.



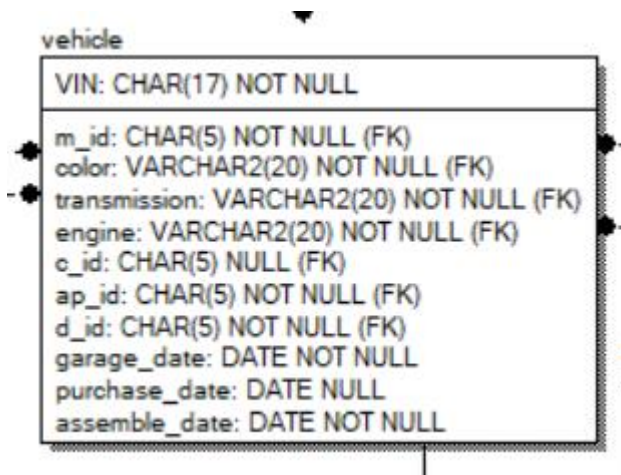
### 3.3 model



m\_id는 char(5)로 5자리의 id값을 가지는데, 8XXXX의 값을 가질 수 있도록 constraint를 설정하였다.

name, price, body\_style 모두 null이 들어오지 못 하도록 하였고, address를null 값을 허용하지 않은 이유는 차가 조립되면 차고로 보내야되는데 그 때 dealer 차고의 주소가 필요하기 때문이다 model의 relation은 아래와 같이 나타낼 수 있다.

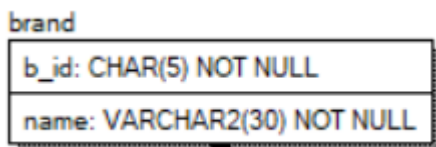
### 3.4 vehicle



VIN은 char(17)로 17자리 수의 값으로 구성된다. null 값을 가질 수 없다. m\_id는 char(5)로 null값을 가질 수 없다.color, transmission, engine은 모두 varchar(20)이며 option이므로 null 값을 가질 수 없다. c\_id는 char(5)로 차가 아직 팔리지 않은 상태 라면 c\_id가 null 값으로 들어올 수 있다. ap\_id, d\_id는 char(5)로 null 값을 가질 수 없다.

garage\_date, assemble\_date 또한 null 값을 허용하지 않지만, purchase\_date는 팔리지 않은 상태라면 c\_id와 마찬가지로 null 값을 가질 수 있다 위 날짜를 나타내는 attribute는 date type이다.

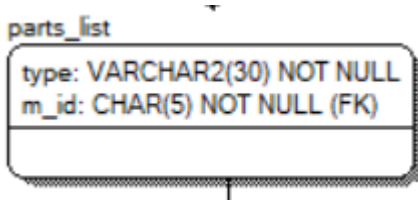
### 3.5 brand



b\_id는 char(5)로 5자리의 id값을 가지는데, 1XXXX의 값을 가질 수 있도록 constraint를 설정하였다.

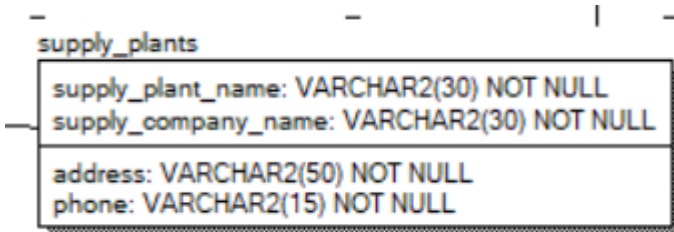
name은 brand의 이름이므로 null이 들어오지 못 하도록 하였고, varchar(30)으로 설정하였다.

### 3.6 parts\_list



m\_id는 char(5)로 5자리의 id값을 가지고, type은 varchar(30)으로 둘 다 null 값을 허용하지 않았다.

### 3.7 supply\_plants

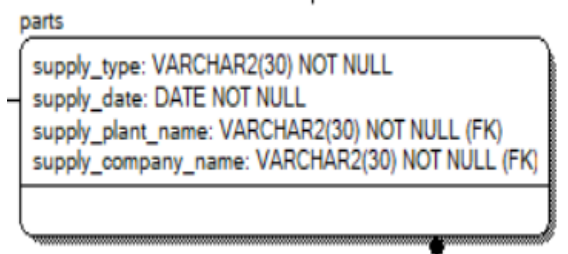


supply_plants
supply_plant_name: VARCHAR2(30) NOT NULL
supply_company_name: VARCHAR2(30) NOT NULL
address: VARCHAR2(50) NOT NULL
phone: VARCHAR2(15) NOT NULL

supply\_plant\_name과 supply\_company\_name은 varchar(30)으로 설정하였고, PK이므로 null 값이 허용되지 않는다. address와 phone은 각각

varchar(50), varchar(15)으로 하였고, null 값을 허용하지 않았다.

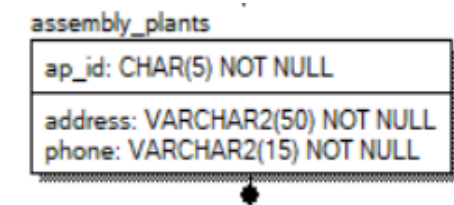
### 3.8 parts



parts
supply_type: VARCHAR2(30) NOT NULL
supply_date: DATE NOT NULL
supply_plant_name: VARCHAR2(30) NOT NULL (FK)
supply_company_name: VARCHAR2(30) NOT NULL (FK)

supply\_plant\_name과 supply\_company\_name은 varchar(30)으로 설정하였고, PK이므로 null 값이 허용되지 않는다. supply\_type과 supply\_date는 각각 varchar(30), date로 하였고, null 값을 허용하지 않았다.

### 3.9 assembly\_plants

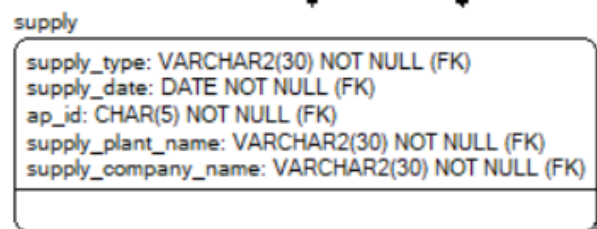


assembly_plants
ap_id: CHAR(5) NOT NULL
address: VARCHAR2(50) NOT NULL
phone: VARCHAR2(15) NOT NULL

ap\_id는 char(5)로 5자리의 id값을 가지는데, 9XXXX의 값을 가질 수 있도록 constraint를 설정하였다.

address와 phone은 null이 들어오지 못 하도록 하였고, 각각 varchar(50), varchar(15)으로 설정하였다.

### 3.10 supply



supply
supply_type: VARCHAR2(30) NOT NULL (FK)
supply_date: DATE NOT NULL (FK)
ap_id: CHAR(5) NOT NULL (FK)
supply_plant_name: VARCHAR2(30) NOT NULL (FK)
supply_company_name: VARCHAR2(30) NOT NULL (FK)

supply\_plant\_name과 supply\_company\_name, supply\_type은 varchar(30)으로 설정하였고, null 값이 허용되지 않는다. supply\_date는 date로 하였고, null 값을 허용하지 않았다. ap\_id는 char(5)로 설정하였고, null 값을 허용할 수 없다.

### 3.11 consist

consist

```
VIN: CHAR(17) NOT NULL (FK)
supply_type: VARCHAR2(30) NOT NULL (FK)
supply_date: DATE NOT NULL (FK)
supply_plant_name: VARCHAR2(30) NOT NULL (FK)
supply_company_name: VARCHAR2(30) NOT NULL (FK)
```

supply\_plant\_name과 supply\_company\_name, supply\_type은 varchar(30)으로 설정하였고, null 값이 허용되지 않는다. supply\_date는 date로 하였고, null 값을 허용하지 않았다. VIN은 char(17)로 설정하였고, null 값을 허용할 수 없다.

### 3.12 out\_sourcing

out\_sourcing

```
supply_plant_name: VARCHAR2(30) NOT NULL (FK)
supply_company_name: VARCHAR2(30) NOT NULL (FK)
parts_type: VARCHAR2(30) NOT NULL (FK)
m_id: CHAR(5) NOT NULL (FK)
```

supply\_plant\_name과 supply\_company\_name, parts\_type은 varchar(30)으로 설정하였고, null 값이 허용되지 않는다. supply\_date는 date로 하였고, null 값을 허용하지 않았다. m\_id는 char(5)로 설정하였고, null 값을 허용할 수 없다

### 3.13 model\_price

model\_price

```
engine: VARCHAR2(20) NOT NULL
color: VARCHAR2(20) NOT NULL
transmission: VARCHAR2(20) NOT NULL
m_id: CHAR(5) NOT NULL (FK)
price: NUMERIC(9,0) NOT NULL
```

color, transmission, engine은 모두 varchar(20)이며 option이므로 null 값을 가질 수 없다. m\_id는 char(5)로 model없는 option없기 때문에 null 값을 허용하지 않는다. price는 numeric(9,0)으로 설정하였고, 차의 가격이므로 null 값을 허용하지 않았다.

## 4. MySQL

테이블을 생성하는 create, 튜플을 삽입하는 insert, 프로그램 종료시 테이블을 없애는 drop으로 구성되어있다. 제약조건에 맞게 설정한 SQL create 코드는 다음과 같다.

```
create table brand
(b_id          char(5),
name          varchar(30) not null,
primary key(b_id),
check(b_id >='10000' and b_id <'20000'))
);

create table model
(m_id      char(5),
b_id      char(5) not null,
name      varchar(20) not null,
price     numeric(9,0) not null ,
body_style varchar(30) not null,
primary key(m_id),
foreign key(b_id) references brand (b_id)
on delete cascade,
check(m_id >='80000' and m_id <'90000'))
);

create table model_price
(m_id      char(5),
engine varchar(20) ,
color     varchar(20) ,
transmission      varchar(20) ,
price     numeric(9,0),
primary key (engine,color,transmission,m_id),
foreign key (m_id) references model (m_id)
on delete cascade
);

create table customer
(c_id      char(5),
name varchar(10) not null,
address   varchar(50),
phone     varchar(15) not null,
gender    char not null,
annual_income      numeric(10,2),
check (annual_income > 0) ,
```

```

primary key (c_id),
check(c_id>='20000' and c_id < '30000')
);

create table dealer
(d_id          char(5),
name          varchar(10) not null,
address       varchar(50) not null,
phone         varchar(15) not null,
primary key(d_id),
check('60000' <= d_id and d_id < '70000')
);

create table assembly_plants
(ap_id        char(5),
address       varchar(50) not null,
phone        varchar(15) not null,
primary key (ap_id),
check('90000' <= ap_id and ap_id <='99999')
);

create table supply_plants
(phone        varchar(15) not null,
address       varchar(50) not null,
supply_plant_name          varchar(30),
supply_company_name        varchar(30) ,
primary key (supply_plant_name,supply_company_name)
);

create table parts
(supply_type          varchar(30),
supply_date          date,
supply_plant_name          varchar(30),
supply_company_name        varchar(30),
primary key (supply_type,supply_date,supply_plant_name,supply_company_name),
foreign key (supply_plant_name,supply_company_name) references supply_plants (supply_plant_name,supply_company_name)
on delete cascade
);

create table supply
(ap_id          char(5),
supply_type          varchar(30),
supply_date          date,
supply_plant_name          varchar(30),

```

```

supply_company_name          varchar(30),
primary key (ap_id,supply_type,supply_date,supply_plant_name,supply_company_name),
foreign key (ap_id) references assembly_plants (ap_id)
on delete cascade,
foreign key (supply_type,supply_date,supply_plant_name,supply_company_name) references parts
(supply_type,supply_date,supply_plant_name,supply_company_name)
on delete cascade
);

```

```

create table parts_list
(parts_type          varchar(20),
m_id                char(5),
primary key (parts_type,m_id),
foreign key (m_id) references model (m_id)
on delete cascade
);

```

```

create table out_sourcing
(parts_type          varchar(20),
m_id                char(5) ,
supply_plant_name    varchar(30),
supply_company_name  varchar(30),
primary key (parts_type,m_id,supply_plant_name,supply_company_name),
foreign key (supply_plant_name,supply_company_name) references supply_plants (supply_plant_name,supply_company_name)
on delete cascade,
foreign key (parts_type,m_id) references parts_list (parts_type,m_id)
on delete cascade
);

```

```

create table vehicle
(VIN                  char(17),
garage_date           date,
purchase_date         date,
assemble_date         date,
c_id                  char(5),
d_id                  char(5),
ap_id                 char(5),
m_id                  char(5),
engine                varchar(20) ,
color                 varchar(20) ,
transmission          varchar(20) ,
primary key (VIN),
foreign key (c_id) references customer (c_id)
on delete set null,

```

```

foreign key (m_id) references model (m_id)
on delete set null,
foreign key (ap_id) references assembly_plants (ap_id)
on delete set null,
foreign key (d_id) references dealer (d_id)
on delete set null,
foreign key (engine, color, transmission) references model_price (engine, color, transmission)
on delete set null
);
create table consist
(VIN          char(17) ,
supply_type  varchar(30),
supply_date  date,
supply_plant_name  varchar(30),
supply_company_name varchar(30),
primary key (VIN,supply_type,supply_date,supply_plant_name,supply_company_name),
foreign key (VIN) references vehicle (VIN)
on delete cascade,
foreign key (supply_type,supply_date,supply_plant_name,supply_company_name) references parts
(supply_type,supply_date,supply_plant_name,supply_company_name)
on delete cascade
);

```

insert를 통해 삽입한 데이터는 다음과 같다. 마지막으로 프로그램 종료 시 delete로 insert된 date를 모두 제거한 다음, 모든 테이블을 drop하였다.

VIN	garage_date	purchase_date	assemble_date	c_id	d_id	ap_id	m_id	engine	color	transmission
100000000000000000	2019-11-05	2019-12-29	2019-11-01	20000	60000	90000	80100	v1	grey	2
100000000000000001	2020-12-11	2020-12-13	2020-12-06	20001	60001	90000	80101	v1	grey	2
100000000000000002	2020-03-20	2020-04-01	2020-03-16	20002	60002	90000	80102	v1	black	3
100000000000000003	2019-12-30	2020-01-12	2019-12-26	20003	60003	90000	80103	v2	black	2
100000000000000004	2019-11-14	2020-01-05	2019-11-11	20004	60004	90000	80104	v2	white	1
100000000000000005	2020-04-17	2020-08-17	2020-04-13	20005	60005	90000	80105	v2	grey	1
100000000000000006	2021-05-13	2021-06-05	2021-05-09	20006	60006	90000	80106	v3	grey	2
100000000000000007	2020-08-21	2020-11-09	2020-08-17	20007	60007	90000	80107	v3	white	2
100000000000000008	2020-02-10	2020-03-13	2020-01-18	20015	60000	90000	80113	v5	skyblue	3
100000000000000009	2019-11-05	2020-02-19	2019-11-01	20000	60000	90000	80105	v2	grey	1
100000000000000010	2020-12-11	2021-04-17	2020-12-06	20001	60001	90000	80106	v3	grey	2
100000000000000011	2021-03-20	2021-03-21	2021-03-16	20014	60002	90000	80107	v3	white	2
100000000000000012	2019-12-30	2020-02-18	2019-12-26	20013	60008	90000	80108	v3	grey	3
100000000000000013	2019-11-14	2019-12-19	2019-11-11	20012	60009	90000	80110	v4	black	2
100000000000000014	2020-04-17	2020-06-11	2020-04-13	20011	60010	90000	80110	v4	black	2
100000000000000015	2021-05-13	2021-06-04	2021-05-09	20010	60011	90000	80110	v4	black	2
100000000000000016	2021-04-21	NULL	2021-03-17	NULL	60012	90000	80102	v1	black	3
100000000000000017	2021-05-10	NULL	2021-04-06	NULL	60013	90000	80103	v2	black	2
100000000000000017	2021-02-27	NULL	2021-01-07	NULL	60004	90000	80104	v2	white	1

(vehicle의 예시)

SET SQL\_SAFE\_UPDATES = 0;

delete from consist;

delete from vehicle;

delete from out\_sourcing;

delete from parts\_list;

delete from supply;

delete from parts;

delete from supply\_plants;

delete from assembly\_plants;

delete from dealer;

delete from customer;

delete from model\_price;

delete from model;

delete from brand;

drop table consist;

drop table vehicle;

drop table out\_sourcing;

drop table parts\_list;

drop table supply;

drop table parts;

drop table supply\_plants;

drop table assembly\_plants;

drop table dealer;

drop table customer;

drop table model\_price;

drop table model;

drop table brand;

위 과정은 table을 create한 순서 반대로 진행한다.



## 5. ODBC C language codes

```
fp=fopen("20171682_1.txt","r"); //create table & insert values
fseek(fp,0,SEEK_END);
size=ftell(fp);
buffer=malloc((double)size+1);
memset(buffer,0,(double)size+1);
fseek(fp,0,SEEK_SET);
fread(buffer,size,1,fp);

temp=strtok(buffer,";");
while(1){
    if(temp==NULL)break;
    sprintf(query,"%s",temp);
    state=mysql_query(connection,query);
    temp=strtok(NULL,";");
}
fclose(fp);
```

20171682\_1.txt에는 table을 create하는 MySQL 코드와 values를 insert하는 mysql 코드가 들어있다. 그 파일의 크기만큼 buffer를 동적할당하여 읽는다. 읽은 sql코드를 ;를 기준으로 구분한 뒤 mysql\_query()를 이용하여 table을 create하고 data들을 insert 한다.

```
if(type==0){
    fp=fopen("20171682_2.txt","r");
    fseek(fp,0,SEEK_END);
    size=ftell(fp);
    buffer=malloc((double)size+1);
    memset(buffer,0,(double)size+1);
    fseek(fp,0,SEEK_SET);
    fread(buffer,size,1,fp);
    temp=strtok(buffer,";");
    while(1){
        if(temp==NULL)break;
        sprintf(query,"%s",temp);
        state=mysql_query(connection,query);
        temp=strtok(NULL,";");
    }
    fclose(fp);
    break;
}
```

prompt창에서 0이 입력된 경우, 사용했던 data와 table을 모두 delete하고 종료해야한다. tuple들을 delete하는 코드와 table을 drop하는 코드들이 20171682\_2.txt에 저장되어 있다. 역시 ; 기준으로 parsing하며 mysql\_query()를 이용하여 delete와 drop을 수행하고 종료한다.

query문을 실행하는 부분은 다음과 같은 형식으로 진행된다.

```
printf("** Show the sales trends for a particular brand over the past k years**\n");

printf(" Which brand? : ");
fgets(command, 100, stdin);
command[strlen(command) - 1] = 0;
printf(" Which K? : ");
scanf("%d", &K);
getchar();

sprintf(temp2, "create view bid as select b_id from brand where
name = W"%sW"; select year(purchase_date) as year, count(*) as sales    from( select * from
model natural join bid natural join vehicle where year(purchase_date) >= year(date_sub(NOW(),
interval %d year))) as S group by year(S.purchase_date)", command, K);
```

prompt에 각 TYPE마다 출력할 부분을 처리한 다음, 입력이 필요하면 받는다. 받고 sprintf()를 사용하여 sql 쿼리문이 담긴 문자열을 변수에 저장한다.

```
> temp = strtok(temp2, ";");
> printf("year\t\tsales\n");
> while (1) {
>     if (temp == NULL) {
>         break;
>     }
>     sprintf(query, "%s", temp);
>     state = mysql_query(connection, query);
>     if (state == 0) {
>         sql_result = mysql_store_result(connection);
>         if (sql_result != NULL) {
>             while ((sql_row = mysql_fetch_row(sql_result)) != NULL) {
>                 fields = mysql_num_fields(sql_result);
>                 for (i = 0; i < fields; i++)
>                     printf("%s\t", sql_row[i]);
>                 printf("\n");
>             }
>         }
>         temp = strtok(NULL, ";");
>     }
> }
```

while문으로 돌면서 ; 를 기준으로 parsing하면서 쿼리문이 끝날 때까지 실행한다.

mysql\_query() 함수가 실제 쿼리문을 실행하는 부분이고, mysql\_store\_result()와 mysql\_fetch\_row() 부분이 실행된 결과를 가져오는 부분이다. 다른 TYPE에서도 위와 같은 방식으로 처리한다.

## 6. Queries

**(TYPE 1) Show the sales trends for a particular brand over the past k years.**

```
create view bid as
select b_id
from brand
where name = W"%sW";
select year(purchase_date) as year, count(*) as sales
from( select * from model natural join bid natural join vehicle
where year(purchase_date) >= year(date_sub(NOW(), interval %d year))) as S
group by year(S.purchase_date);
```

```
---- TYPE 1 ----
** Show the sales trends for a particular brand over the past k years**
Which brand? : Audi
Which K? : 2
year      sales
2019      1
2020      2
```

bid라는 view를 만들어서 입력받은 brand name의 b\_id를 구한 다음, vehicle과 natural join을 한다. 그 다음 purchase\_date의 년도와 현재 날짜의 년도를 따지면서 연도별로 출력하였다.

**(TYPE 1-1) Then break these data out by gender of the buyer.**

```
with findk(c_id,date,gender) as
(select T.c_id, purchase_date, gender
from model natural join bid natural join vehicle as T, customer as C
where year(purchase_date) >= year(date_sub(NOW(), interval %d year)) and C.c_id = T.c_id)
select year(date) as year, count(*) as sales, count(case when gender = 'F' then 1 end) as Female,
count(case when gender = 'M' then 1 end) as Male from findk
group by year(date);
```

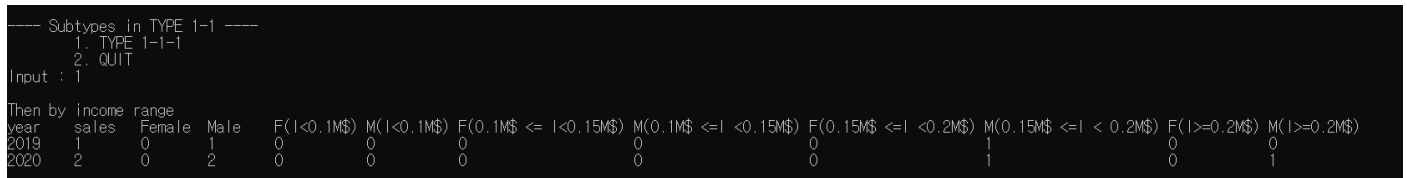
```
---- Subtypes in TYPE 1 ----
1. TYPE 1-1
2. QUIT
Input : 1

Then break these data out by gender of the buyer
year      sales   Female   Male
2019      1       0        1
2020      2       0        2
```

TYPE1의 정보를 출력한 뒤, 추가 입력을 받아 1-1의 내용을 볼 지 선택하게 하였고, 선택하면 TYPE1에서 구한 정보를 성별로 구분하여 그 인원 수를 출력하였다.

### (TYPE 1-1-1) Then by income range.

```
with findk(c_id,date,gender,income) as
(select T.c_id, purchase_date, gender, annual_income
from model natural join bid natural join vehicle as T, customer as C
where year(purchase_date) >= year(date_sub(NOW(), interval %d year)) and C.c_id = T.c_id )
select year(date) as year, count(*) as sales, count(case when gender = 'F' then 1 end) as Female,
count(case when gender = 'M' then 1 end) as Male,
count(case when income < 100000 and gender = 'F' then 1 end) as 'Female(income < 100000)',
count(case when income < 100000 and gender = 'M' then 1 end),
count(case when 100000 <= income and income < 150000 and gender = 'F' then 1 end),
count(case when 100000 <= income and income < 150000 and gender = 'M' then 1 end) ,
count(case when 150000 <= income and income < 200000 and gender = 'F' then 1 end) ,
count(case when 150000 <= income and income < 200000 and gender = 'M' then 1 end),
count(case when income >= 200000 and gender = 'F' then 1 end),
count(case when income >= 200000 and gender = 'M' then 1 end)
from findk
group by year(date);
```

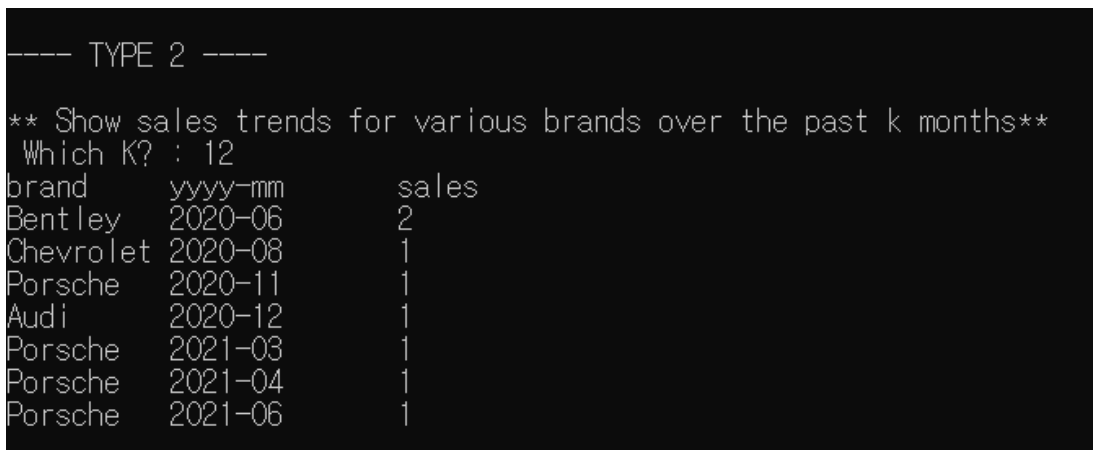


Subtypes in TYPE 1-1											
1. TYPE 1-1-1											
2. QUIT											
Input : 1											
Then by	income	range									
year	sales	Female	Male	F(I<0.1M\$)	M(I<0.1M\$)	F(0.1M\$ <= I < 0.15M\$)	M(0.1M\$ <= I < 0.15M\$)	F(0.15M\$ <= I < 0.2M\$)	M(0.15M\$ <= I < 0.2M\$)	F(I>=0.2M\$)	M(I>=0.2M\$)
2019	1	0	1	0	0	0	0	0	0	0	0
2020	2	0	2	0	0	0	0	1	1	0	1

TYPE 1-1의 결과가 출력되면, 추가로 1-1-1의 내용을 볼 지 선택하게끔 입력을 받아, 선택하면 TYPE 1-1에서 구한 정보를 성별 별로 income의 range를 두어서, 그 인원 수를 출력하였다.

### (TYPE 2) Show sales trends for various brands over the past k months.

```
with temp(bname,mid,date) as
(select B.name as bname, M.m_id as mid, purchase_date
from model as M, brand as B natural join vehicle as T
where M.b_id = B.b_id and M.m_id = T.m_id and T.c_id is not null
and date_format(purchase_date,"%Y-%m") >= date_format(date_sub(NOW(), interval %d month), "%Y-%m")) select
bname, date_format(date, "%Y-%m") as "year-month", count(*) as sales
from temp
group by bname, month(date)
order by date;
```



TYPE 2		
** Show sales trends for various brands over the past k months**		
Which K? : 12		
brand	yyyy-mm	sales
Bentley	2020-06	2
Chevrolet	2020-08	1
Porsche	2020-11	1
Audi	2020-12	1
Porsche	2021-03	1
Porsche	2021-04	1
Porsche	2021-06	1

현재 날짜의 월에서 K month만큼 뺀 다음 date\_format을 비교하여 값을 얻었다.

**(TYPE 2-1) Then break these data out by gender of the buyer.**

```
with temp(bname,mid,date,gender) as
(select B.name as bname, M.m_id as mid, purchase_date, gender
from model as M, brand as B natural join vehicle as T, customer as C
where M.b_id = B.b_id and M.m_id = T.m_id and T.c_id = C.c_id
and date_format(purchase_date,"%Y-%m") >= date_format(date_sub(NOW(), interval %d month), "%Y-%m")) select
bname, date_format(date, "%Y-%m") as "year-month", count(*) as sales,
count(case when gender = 'F' then 1 end) as Female,
count(case when gender = 'M' then 1 end) as Male
from temp
group by bname, month(date)
order by date;
```

```
---- Subtypes in TYPE 2 ----
      1. TYPE 2-1
      2. QUIT
Input : 1

Then break these data out by gender of the buyer
brand      yyyy-mm      sales  Female  Male
Bentley    2020-06        2        2        0
Chevrolet  2020-08        1        0        1
Porsche    2020-11        1        0        1
Audi       2020-12        1        0        1
Porsche    2021-03        1        1        0
Porsche    2021-04        1        0        1
Porsche    2021-06        1        0        1
```

TYPE 2의 결과가 출력되면, 추가로 2-1의 내용을 볼 지 선택하게끔 입력을 받아, 선택하면 TYPE 2에서 구한 정보를 성별에 따라 구분하였다.

**(TYPE 2-1-1) Then by income range.**

```
with temp(bname,mid,date,gender,income) as
(select B.name as bname, M.m_id as mid, purchase_date, gender, annual_income
from model as M, brand as B natural join vehicle as T, customer as C
where M.b_id = B.b_id and M.m_id = T.m_id and T.c_id = C.c_id
and date_format(purchase_date,"%%Y-%%m%%W") >= date_format(date_sub(NOW(), interval %d
month),"%%Y-%%m%%W"))
select bname, date_format(date, "%%Y-%%m%%W") as "year-monthW", count(*) as sales,
count(case when gender = 'F' then 1 end) as Female,
count(case when gender = 'M' then 1 end) as Male,
count(case when income < 100000 and gender = 'F' then 1 end) as 'Female(income < 100000)',
count(case when income < 100000 and gender = 'M' then 1 end),
count(case when 100000 <= income and income < 150000 and gender = 'F' then 1 end),
count(case when 100000 <= income and income < 150000 and gender = 'M' then 1 end),
count(case when 150000 <= income and income < 200000 and gender = 'F' then 1 end),
count(case when 150000 <= income and income < 200000 and gender = 'M' then 1 end),
count(case when income >= 200000 and gender = 'F' then 1 end),
count(case when income >= 200000 and gender = 'M' then 1 end)
from temp
group by bname, month(date)
by date;
```

```

---- Subtypes in TYPE 2-1 ----
1. TYPE 2-1-1
2. QUIT
Input : 1
Then by income range
brand      yyyy-mm    sales  Female  Male  F(1<0.1M$) M(1<0.1M$) F(0.1M$ <= 1<0.15M$) M(0.1M$ <=1 <0.15M$) F(0.15M$ <=1 <0.2M$) M(0.15M$ <=1 < 0.2M$) F(1>=0.2M$) M(1>=0.2M$)
Bentley    2020-06        2       2       0       0       0       2       0       0       0       0       0
Chevrolet  2020-08        1       0       1       0       0       0       1       0       0       0
Porsche    2020-11        1       0       1       0       0       0       1       0       0       0
Audi       2020-12        1       0       1       0       0       0       1       0       0       0
Porsche    2021-03        1       1       0       0       0       1       0       0       0       0
Porsche    2021-04        1       0       1       0       0       0       0       1       0       0
Porsche    2021-06        1       0       1       0       0       0       1       0       0       0

```

TYPE 2-1의 결과가 출력되면, 추가로 2-1-1의 내용을 볼 지 선택하게끔 입력을 받아, 선택하면 TYPE 2-1에서 구한 정보를 다시 성별과 income에 따라 구분하였다.

**(TYPE 3) Find that transmissions made by supplier (company name) between two given dates are defective.**

```

select supply_date, supply_plant_name, supply_company_name
from consist where supply_company_name = '%s'
and supply_date between W"%d-%d-%dW" and W"%d-%d-%dW" and supply_type = W"transmissionW";

```

```

---- TYPE 3 ----

** Find that transmissions made by supplier (company name) between two given dates are defective **
Date from (YYYYMMDD)? : 20181123
Date to   (YYYYMMDD)? : 20190324
Supplier  ? : Korea
supply_date supply_plant_name supply_company_name
2018-12-17 korp2             Korea
2018-12-17 korp2             Korea
2018-12-17 korp2             Korea

```

입력받은 두 날짜와 공급회사 이름을 입력 받아 결함이 있는 transmission의 정보를 출력하였다.

**(TYPE 3-1) Find the VIN of each car containing such a transmission and the customer to which it was sold.**

```

select supply_date, supply_plant_name, supply_company_name,VIN,C.name as customer
from consist natural join Vehicle as V, customer as C
where C.c_id = V.c_id and supply_company_name = '%s'
and supply_date between W"%d-%d-%dW" and W"%d-%d-%dW"
and supply_type = W"transmissionW";

```

```

---- Subtypes in TYPE 3 ----
1. TYPE 3-1
2. TYPE 3-2
3. QUIT
Input : 1

Find the VIN of each car containing such a transmission and the customer to which it was sold
supply_date supply_plant_name supply_company_name VIN customer
2018-12-17 korp2             Korea          100000000000000013 YeongLan
2018-12-17 korp2             Korea          100000000000000014 MyeongHwa
2018-12-17 korp2             Korea          100000000000000015 HyeJi

```

TYPE 3의 내용이 출력이 되면, 추가로 입력을 받아 TYPE 3-1, 3-2 둘 중 하나를 선택하게끔 구현하였다.

TYPE 3-1을 선택하면 TYPE 3에서 입력받은 내용을 토대로 결함이 있던 transmission이 어떤 차에 사용되었고, 그 차를 구매한 고객의 이름을 출력하였다.

**(TYPE 3-2) Find the dealer who sold the VIN and transmission for each vehicle containing these transmissions.**

```
select supply_date, supply_plant_name, supply_company_name, name as dealer
from consist natural join Vehicle natural join dealer
where supply_company_name = '%s' and supply_date between W"%d-%d-%dW" and W"%d-%d-%dW"
and supply_type = W"transmissionW";
```

----- Subtypes in TYPE 3 -----

1. TYPE 3-1
2. TYPE 3-2
3. QUIT

Input : 2

Find the dealer who sold the VIN and transmission for each vehicle containing these transmissions

supply_date	supply_plant_name	supply_company_name	dealer
2018-12-17	korp2	Korea	Alex
2018-12-17	korp2	Korea	Kasar
2018-12-17	korp2	Korea	Issac

TYPE 3에서 입력받은 내용을 토대로 결함이 있던 transmission이 어떤 dealer에게 팔렸는지 알 수 있도록 딜러의 이름을 출력하였다.

**(TYPE 4) Find the top k brands by dollar-amount sold by the year.**

```
create view temp as
(select B.name as bname, M.m_id as mid, purchase_date, engine, color, transmission
from model as M, brand as B natural join vehicle as T where M.b_id = B.b_id and M.m_id = T.m_id
and T.c_id is not null );
with SALES(date, bname, price) as (select purchase_date, bname, price
from temp natural join model_price where mid = m_id ),
dollar(year, bname, sales) as
(select year(date) as year, bname, sum(price) as dollar_amount
from SALES group by year(date), price order by year(date)),
order_val as
( SELECT year, bname, sales, RANK() OVER( PARTITION BY year ORDER BY sales desc ) tmp_rank
FROM dollar )
select*
from order_val
WHERE tmp_rank <= %d;
```

----- TYPE 4 -----

\*\* Find the top k brands by dollar-amount sold by the year\*\*

Which K? : 4

year	brand	sales	rank
2019	Audi	53000	1
2020	Bugatti	3330000	1
2020	Audi	285000	2
2020	Porsche	156300	3
2020	Audi	154000	4
2021	Porsche	170800	1
2021	Porsche	147800	2

각 연도별로 dollar-amount기준 1등부터 k등까지 출력되게 하였다.

**(TYPE 5) Find the top k brands by unit sales by the year.**

```
create view temp as
(select B.name as bname, M.m_id as mid, purchase_date, engine, color, transmission
from model as M, brand as B natural join vehicle as T
where M.b_id = B.b_id and M.m_id = T.m_id and T.c_id is not null );
with sales as
(select year(purchase_date) as year, bname, count(bname) as unit_sales
from temp group by year(purchase_date), bname),
order_val as( SELECT year, bname, unit_sales, RANK() OVER( PARTITION BY year ORDER BY unit_sales desc )
tmp_rank FROM sales )
select year, bname, unit_sales, tmp_rank as ranking
from order_val WHERE tmp_rank <= %d;
```

```
---- TYPE 5 ----

** Find the top k brands by unit sales by the year**
Which K? : 4
year brand      sales    rank
2019 Audi       1         1
2019 Bentley    1         1
2020 Chevrolet  4         1
2020 Audi       2         2
2020 Porsche    2         2
2020 Bugatti    1         4
2020 Bentley    1         4
2021 Porsche    3         1
2021 Bentley    1         2
```

각 연도별로 unit sales 기준 1등부터 k등까지 출력하였다.

**(TYPE 6) In what month(s) do convertibles sell best?**

```
with conv(date) as ( select purchase_date
from vehicle natural join model
where c_id is not null and body_style = 'Convertible' ),
temp(months, count) as( select date_format(date, W"%MMW") as best_sold_months, count(date)
from conv group by month(date) order by count(date) desc )
select months as best_sold_month
from temp where count >= (select max(count) from temp);
```

```
---- TYPE 6 ----

** In what month(s) do convertibles sell best?**
-----
best_sold_month
June
```

**(TYPE 7) Find those dealers who keep a vehicle in inventory for the longest average time.**

```
with temp(did,cnt1,cnt2) as
( select d_id, avg((case when purchase_date is null then datediff(now(), garage_date) end)) as t1,
avg((case when purchase_date is not null then datediff(purchase_date, garage_date) end)) as t2
from vehicle group by d_id),
temp2(did, keep) as( select did,
ifnull((case when cnt1 is not null and cnt2 is not null then(cnt1 + cnt2) / 2 end),
ifnull(cnt1, cnt2)) as t1 from temp)
select name as longest_dealer, keep as kept_time
from temp2, dealer where did = d_id and keep >= (select max(keep) from temp2);
```



----- TYPE 7 -----

\*\* Find those dealers who keep a vehicle in inventory for the longest average time\*\*

-----  
dealer Kept\_time  
dooon 118.50000000