

과목명: 데이터베이스시스템

담당 교수명: 정성원

<<Project 2>>

서강대학교 컴퓨터학과

[20171630]

[남주형]

1. 프로젝트 개요

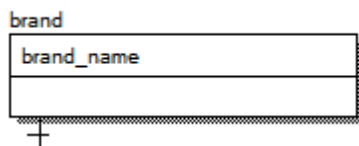
프로젝트 1에서 설계한 스키마를 BCNF를 만족하도록 수정하고 수정된 스키마를 바탕으로 DB에 실제 데이터를 넣어보고 쿼리를 처리하는 소규모 관계형 데이터 베이스를 구축한다.

2. BCNF Decomposition

※project 1에서 만든 스키마를 이용하여 bcnf를 검사하였다.

※attribute의 순서에따라 A~Z 기호로 표시

1) brand

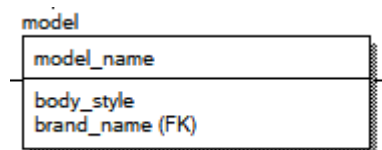


Function Dependencies: -

Superkey: A

BCNF: 만족

2) model

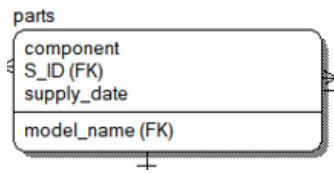


Function Dependencies: A -> BC

Superkey: A

BCNF: 만족

3) parts

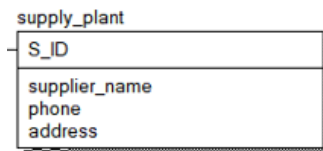


Function Dependencies: ABC → D

Superkey: ABC

BCNF: 만족

4) supply_plant

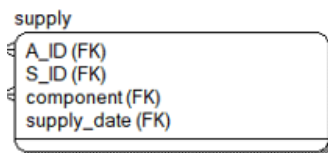


Function Dependencies: A → BCD

Superkey: A

BCNF: 만족

5) supply

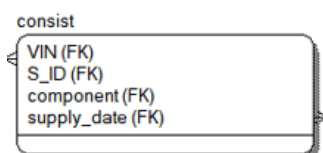


Function Dependencies: -

Superkey: ABCD

BCNF: 만족

6) consist

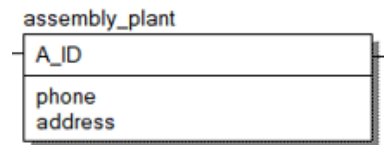


Function Dependencies: -

Superkey: ABCD

BCNF: 만족

7) assembly_plant

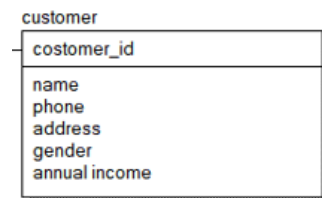


Function Dependencies: A -> BC

Superkey: A

BCNF: 만족

8) customer

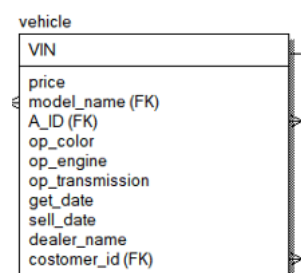


Function Dependencies: A -> BCDEF, C -> ABDEF

Superkey: A, C

BCNF: 만족

9) vehicle



Function Dependencies: A -> BCDEFGHIJ, CEFG -> B

Superkey: A

BCNF: 불만족

Decomposition:

CEFG -> B 의 관계가 trivial하지 않고 CEFG가 vehicle의 superkey가 아니므로 BCNF가 아니다.

따라서 R1(CEFGB), R2(ACDEFGHIJ)로 분해할 수 있다.

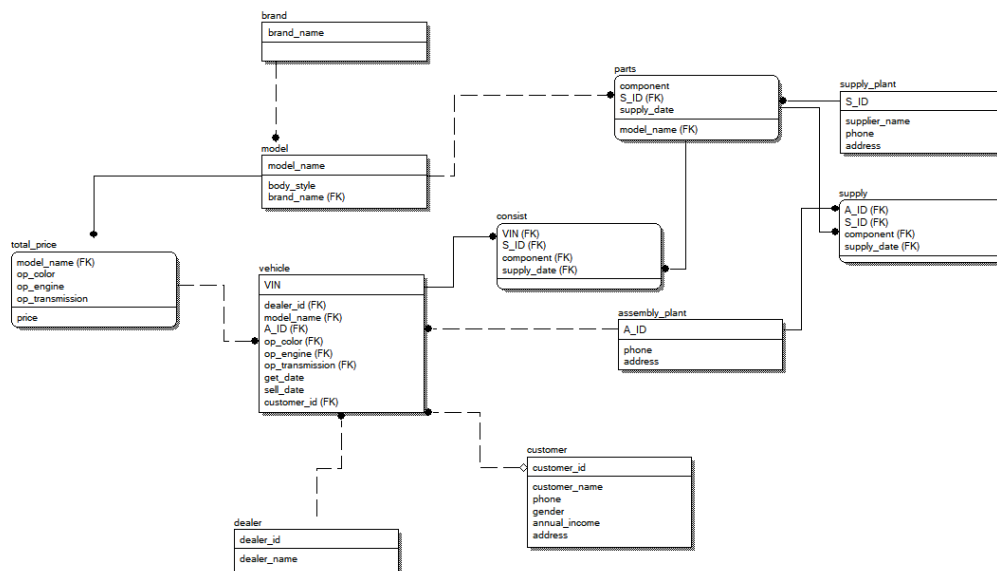
R1은 CEFG->B의 관계에서 CEFG가 R1의 superkey이므로 BCNF를 만족한다.

R2는 A->CDEFGHIJ의 관계에서 A가 R2의 superkey이므로 BCNF를 만족한다.

3. Schema diagram description(erwin-physical mode)

※BCNF Decomposition을 수행한 후 만들어진 physical schema diagram이다.

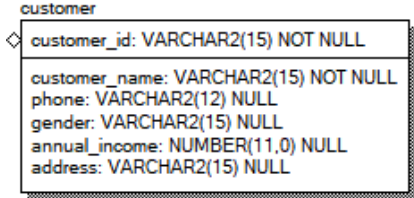
※project 2에서 project 1의 내용을 수정하여 dealer relation을 추가해주었다.



Relation	Description
<p>brand</p> <pre> brand_name: VARCHAR2(15) NOT NULL </pre>	<p>brand_name : 문자형이고 primary key이므로 null값을 가지면 안된다.</p>
<p>model</p> <pre> model_name: VARCHAR2(15) NOT NULL body_style: VARCHAR2(15) NOT NULL brand_name: VARCHAR2(15) NOT NULL (FK) </pre>	<p>model_name: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>body_style : 문자형이고 차량마다 style이 있으므로 null값을 가지면 안된다.</p> <p>brand_name : 문자형이고 model이 brand에 속해 있으므로 null값을 가지면 안된다.</p>
<p>total_price</p> <pre> model_name: VARCHAR2(15) NOT NULL (FK) op_color: VARCHAR2(15) NOT NULL op_engine: VARCHAR2(15) NOT NULL op_transmission: VARCHAR2(15) NOT NULL price: NUMBER(10,0) NULL </pre>	<p>model_name: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>op_color: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>op_engine: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>op_transmission: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>price: number형이고 가격은 무조건 존재해야하므로 null값을 가지면 안된다. 또한 0원이상의 가격이어야 하므로 price>=0의 constraint를 가진다.</p>
	<p>VIN: 17자리로 이루어진 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>dealer_id: 문자형이고 null값을 가지면 안된다.</p> <p>model_name: 문자형이고 null값을 가지면 안된다.</p>

<p>vehicle</p> <pre> VIN: VARCHAR2(17) NOT NULL dealer_id: VARCHAR2(15) NOT NULL (FK) model_name: VARCHAR2(15) NOT NULL (FK) A_ID: VARCHAR2(15) NOT NULL (FK) op_color: VARCHAR2(15) NOT NULL (FK) op_engine: VARCHAR2(15) NOT NULL (FK) op_transmission: VARCHAR2(15) NOT NULL (FK) get_date: DATE NOT NULL sell_date: DATE NULL customer_id: VARCHAR2(15) NULL (FK) </pre>	<p>A_ID: 문자형이고 null값을 가지면 안된다.</p> <p>op_color: 문자형이고 null값을 가지면 안된다.</p> <p>op_engine: 문자형이고 null값을 가지면 안된다.</p> <p>op_transmission: 문자형이고 null값을 가지면 안된다.</p> <p>get_date: 날짜를 저장할 수 있는 DATE형이고 null값을 가지면 안된다.</p> <p>sell_date: 날짜를 저장할 수 있는 DATE형이고 아직 팔리지 않은 차량의 경우 null값을 가질 수 있다.</p> <p>customer_id: 문자형이고 아직 팔리지 않은 차량의 경우 고객이 없으므로 null값을 가질 수 있다.</p>
<p>dealer</p> <pre> dealer_id: VARCHAR2(15) NOT NULL dealer_name: VARCHAR2(15) NOT NULL </pre>	<p>dealer_id: 문자형이고 primary key이므로 null 값을 가지면 안된다.</p> <p>dealer_name: 문자형이고 null값을 가지면 안된다.</p>
<p>parts</p> <pre> component: VARCHAR2(15) NOT NULL S_ID: VARCHAR2(15) NOT NULL (FK) supply_date: DATE NOT NULL model_name: VARCHAR2(15) NOT NULL (FK) </pre>	<p>component: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>S_ID: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>supply_date: DATE형이고 primary key이므로 null값을 가지면 안된다.</p> <p>model_name: 문자형이고 부품이 특정 모델의 부품이므로 null값을 가지면 안된다.</p>
	<p>VIN: 17자리로 이루어진 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>S_ID: 문자형이고 primary key이므로 null값을</p>

<p>consist</p> <div> VIN: VARCHAR2(17) NOT NULL (FK) S_ID: VARCHAR2(15) NOT NULL (FK) component: VARCHAR2(15) NOT NULL (FK) supply_date: DATE NOT NULL (FK) </div>	<p>가지면 안된다.</p> <p>component: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>supply_date: DATE형이고 primary key이므로 null값을 가지면 안된다.</p>
<p>assembly_plant</p> <div> A_ID: VARCHAR2(15) NOT NULL phone: VARCHAR2(12) NULL address: VARCHAR2(15) NULL </div>	<p>A_ID: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>phone: 문자형이고 전화번호를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p> <p>address: 문자형이고 주소를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p>
<p>supply_plant</p> <div> S_ID: VARCHAR2(15) NOT NULL supplier_name: VARCHAR2(15) NOT NULL phone: VARCHAR2(12) NULL address: VARCHAR2(15) NULL </div>	<p>S_ID: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>supplier_name: 문자형이고 공급공장은 공급업체에 속해 있으므로 null값을 가지면 안된다.</p> <p>phone: 문자형이고 전화번호를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p> <p>address: 문자형이고 주소를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p>
<p>supply</p> <div> A_ID: VARCHAR2(15) NOT NULL (FK) S_ID: VARCHAR2(15) NOT NULL (FK) component: VARCHAR2(15) NOT NULL (FK) supply_date: DATE NOT NULL (FK) </div>	<p>A_ID: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>S_ID: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>component: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>supply_date: DATE형이고 primary key이므로</p>

	null값을 가지면 안된다.
	<p>customer_id: 문자형이고 primary key이므로 null값을 가지면 안된다.</p> <p>customer_name: 문자형이고 null값을 가지면 안된다.</p> <p>phone: 문자형이고 전화번호를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p> <p>gender: 문자형이고 고객이 회사일 수도 있으니 null값을 가져도 된다. 성별을 M(남),F(녀)로 기입하도록 constraint를 가진다.</p> <p>annual_income: number형이고 고객이 회사일 수도 있으니 null값을 가져도 된다. 우리 회사는 연봉 5천만원 이상의 조건부터 고객이 될 수 있다고 설정하였기에 <code>annual_income >= 50000000</code>의 constraint를 가진다.</p> <p>address: 문자형이고 주소를 기입하지 않았을 수도 있으니 null값을 가져도 된다.</p>

4. ODBC C language code

1) ODBC C code의 전체적인 구조

먼저 처음에 20171630_1.txt를 읽어와 table을 create하고 data를 insert한다. 이때 세미콜론을 기준으로 잘라서 `mysql_query` 함수에 넣어준다.

```

/* create and insert */
const char* init = strtok(buffer, ";");
while (init != NULL) {
    state = 0;
    state = mysql_query(connection, init);
    init = strtok(NULL, ";");
}

```

그리고 while문을 통해 query문을 실행한다. 여기서 switch문을 사용하여 각 타입에 맞는 출력을해준다.

또한 각 출력은 각 질문에 맞는 형식으로 설정하여 출력한다.

각 query에 대한 내용은 밑에서 설명하겠다.

0이 선택되면 flag를 1로 설정하고 while문을 빠져나온다.

```
while (1) {
    flag = 0;
    printf("\n\n----- SELECT QUERY TYPES ----- \n\n");
    printf("1. TYPE 1\n");
    printf("2. TYPE 2\n");
    printf("3. TYPE 3\n");
    printf("4. TYPE 4\n");
    printf("5. TYPE 5\n");
    printf("6. TYPE 6\n");
    printf("7. TYPE 7\n");
    printf("0. QUIT\n");
    printf("----- \n");
    printf("Select Number : ");
    scanf("%d", &type);
    getchar();
    switch (type) {
        case 1:
            case 0:
                flag = 1;
                break;
            default:
                flag = 0;
                break;
    }
    if (flag) break;
}
```

'0'이 select 되어 query에 대한 수행이 종료되고 프로그램을 종료하기전에 20171630_2.txt를 읽어 delete와 drop을 실행한다.

```
/* delete and drop */
FILE* fp2 = fopen("20171630_2.txt", "r");
char* buffer2 = NULL;

fseek(fp2, 0, SEEK_END);
size = ftell(fp2);
buffer2 = (char*)malloc(size + 1);
memset(buffer2, 0, size + 1);
fseek(fp2, 0, SEEK_SET);
fread(buffer2, size, 1, fp2);

const char* end = strtok(buffer2, "\n");
while (end != NULL) {
    state = 0;
    state = mysql_query(connection, end);
    end = strtok(NULL, "\n");
}
```

2) query

```
----- 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
-----
```

TYPE 1

k와 brand를 입력받으면 해당 년도에 팔린 차의 개수를 출력해준다. 여기서 k년은 현재 날짜에서 정확히 k년이 아니고 현재 연도가 2021년일 때 k가 3이라면 2019년, 2020년, 2021년의 판매량을 보여준다. 이때 해당 연도에 판매된 이력이 없다면 해당 연도는 생략해주었다.

```

---- TYPE 1 ----

** Show the sales trends for a particular brand over the past k years. **
Which K? : 3
Which brand? : Audi

year  sales
2020  4
2019  2

```

code

```

printf("\n\n");
printf("---- TYPE 1 ----\n\n");
printf("** Show the sales trends for a particular brand over the past k years. **\n");
printf(" Which K? : ");
scanf("%d", &k);
getchar();
printf(" Which brand? : ");
fgets(temp, sizeof(temp), stdin);
temp[strlen(temp) - 1] = '\0';
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name,year(sell_date) as year,customer_id f
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("\nyear  sales\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-4s  %-5s\n", sql_row[0], sql_row[1]);
    }
    mysql_free_result(sql_result);
}

```

mysql query

with trends as

(select brand_name,year(sell_date) as year,customer_id

from vehicle natural join model

where year(sell_date) >= year(NOW() - interval %d year) and brand_name = '%s')

select year, count(year) as sales

from trends

group by year

order by year desc

TYPE 1-1 (TYPE 1을 수행하면 나오는 옵션이다.)

TYPE 1에서 입력받은 k와 brand를 이용해 년도별 팔린 차의 수와 그 수를 남자와 여자로 나누어 보여준다.

TYPE 1에서 보여준 결과를 더 세부적으로 나누어 성별에 따른 판매량을 보여주었다.

```
---- Subtypes in TYPE 1 ----
  1. TYPE 1-1
  0. QUIT
-----
Select Number : 1
---- TYPE 1-1 ----

** Then break these data out by gender of the buyer. **

year  sales  Female  Male
2020   4       0       4
2019   2       2       0
```

code

```
printf("---- Subtypes in TYPE 1 ----\n");
printf("\t1. TYPE 1-1\n");
printf("\t0. QUIT\n");
printf("-----\n");
printf("Select Number : ");
scanf("%d", &subtype);
if (subtype != 1) break;

printf("\n---- TYPE 1-1 ----\n\n");
printf("** Then break these data out by gender of the buyer. **\n");
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name,year(sell_date) as year,customer_id from v");
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("\nyear sales Female Male\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-4s %-5s %-6s %-4s\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3]);
    }
    mysql_free_result(sql_result);
}
printf("\n\n");
```

mysql query

with trends as

(select brand_name,year(sell_date) as year,customer_id

from vehicle natural join model

where year(sell_date) >= year(NOW()) - interval %d year) and brand_name = '%s')

select year, count(year) as sales, count(case when gender = 'F' then 1 end) as Female,count(case
when gender = 'M' then 1 end) as Male

from trends natural join customer

group by year

order by year desc

TYPE 1-1-1 (TYPE 1-1을 수행하면 나오는 옵션이다.)

TYPE 1에서 입력받은 k와 brand를 이용해 년도별 팔린 차의 수와 그 수를 남자와 여자로 나누어 보여준걸 다시 소득별로 나누어준다. TYPE1-1의 subtype이므로 TYPE 1-1에서 보여준 결과를 더 세부적으로 나누어 소득 구간에 따른 차량 판매량을 보여주었다. 이때 각 소득 구간 역시 성별에 따라 나누어서 보여주었다.

```
---- Subtypes in TYPE 1-1 ----
1. TYPE 1-1-1
0. QUIT
-----
Select Number : 1
---- TYPE 1-1-1 ----

** Then by income range. **

year  sales  Female  Male  <F>0~1billion(₩)  <M>0~1billion(₩)  <F>1~2billion(₩)  <M>1~2billion(₩)  <F>2billion(₩)~  <M>2billion(₩)~
2020   4       0       4       0               1               0               1               0               2
2019   2       2       0       0               0               0               0               2               0
```

code

```
printf("\n---- TYPE 1-1-1 ----\n\n");
printf("++ Then by income range. ++\n");
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name, year(sell_date) as year, customer_id from vehicle natural join model where year(sell_date) >= year(NOW()) - interval %d year) and brand_name = '%s') select
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("year sales Female Male <F>0~1billion(₩) <M>0~1billion(₩) <F>1~2billion(₩) <M>1~2billion(₩) <F>2billion(₩)~ <M>2billion(₩)~\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("X-4s X-5s X-6s X-4s X-16s X-16s X-16s X-16s X-15s X-15s\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3], sql_row[4], sql_row[5], sql_row[6], sql_row[7], sql_row[8], sql_row[9]);
    }
    mysql_free_result(sql_result);
}
memset(query, 0, sizeof(query));
```

mysql query

with trends as

(select brand_name, year(sell_date) as year, customer_id from vehicle natural join model
where year(sell_date) >= year(NOW()) - interval %d year) and brand_name = '%s')

select year, count(year) 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 annual_income >= 0 and annual_income < 100000000 and gender = 'F' then 1 end) as
'(F) 0~100000000',

count(case when annual_income >= 0 and annual_income < 100000000 and gender = 'M' then 1 end) as
'(M) 0~100000000',

count(case when annual_income >= 100000000 and annual_income < 200000000 and gender = 'F' then
1 end) as '(F) 0~200000000',

count(case when annual_income >= 100000000 and annual_income < 200000000 and gender = 'M' then

```

1 end) as '(M) 0~200000000',
count(case when annual_income >= 200000000 and gender = 'F' then 1 end) as '(F) 200000000~',
count(case when annual_income >= 200000000 and gender = 'M' then 1 end) as '(M) 200000000~'
from trends natural join customer
group by year
order by year desc

```

TYPE 2

k를 입력받아 이번엔 회사별로 월별 팔린 차의 수를 보여준다. 지난 k 개월의 판매 trend이므로 월 별로 판매량을 보여준다. 여기서 연도가 달라질수 있으므로 년도와 월을 같이 보여준다. 이때도 TYPE 1과 동일하게 정확히 31일전 이런식이 아니라 개월별로 판매량을 보여주었다.

```

---- TYPE 2 ----

** Show sales trends for various brands over the past k months. **
Which K? : 12

brand_name  month      sales
Audi        2020-06    1
Audi        2020-08    1
Audi        2020-09    1
Porsche     2020-08    1
Porsche     2021-03    1
Volkswagen  2020-11    1
Volkswagen  2021-03    1

```

code

```

printf("\n\n");
printf("---- TYPE 2 ----\n\n");
printf("** Show sales trends for various brands over the past k months. **\n");
printf("Which K? : ");
scanf("%d", &k);
getchar();
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name, DATE_FORMAT(sell_date, 'XXYY-MM-DD') as sell_date, sales from trends natural join customer where sell_date <= DATE_ADD(CURDATE(), INTERVAL -%d MONTH))", k);
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("\nbrand_name  month      sales\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-11s  %-7s  %-5s\n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
printf("\n\n");

```

mysql query

```

with trends as
(select brand_name, DATE_FORMAT(sell_date, '%%Y-%%m') as month
from vehicle natural join model
where date_format(sell_date, '%%Y-%%m') >= date_format((NOW() - interval %d month), '%%Y-%%m')
order by brand_name, sell_date)

select*, count(month) as sales
from trends group by brand_name, month
order by brand_name, month

```

TYPE 2-1 (TYPE 2을 수행하면 나오는 옵션이다.)

TYPE 2에서 입력받은 k값을 이용하여 이전 결과에 팔린 개수를 성별에 따라 나누어준다. TYPE 2의 결과를

TYPE 1-1을 처리할 때와 같은 방식으로 보여준다.

```

---- Subtypes in TYPE 2 ----
1. TYPE 2-1
0. QUIT
-----
Select Number : 1
---- TYPE 2-1 ----

** Then break these data out by gender of the buyer. **

brand_name  month  sales  Female  Male
Audi        2020-06  1      0       1
Audi        2020-08  1      0       1
Audi        2020-09  1      0       1
Porsche     2020-08  1      1       0
Porsche     2021-03  1      1       0
Volkswagen  2020-11  1      0       1
Volkswagen  2021-03  1      0       1

```

code

```

printf("---- Subtypes in TYPE 2 ----\n");
printf("1. TYPE 2-1\n");
printf("0. QUIT\n");
printf("-----\n");
printf("Select Number : ");
scanf("%d", &subtype);
if (subtype != 1) break;

printf("\n---- TYPE 2-1 ----\n\n");
printf("** Then break these data out by gender of the buyer. **\n");
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name, DATE_FORMAT(sell_date, '%%Y-%%m') as month, customer_id from
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("\nbrand_name  month  sales  Female  Male\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-11s  %-7s  %-5s  %-6s  %-4s\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3], sql_row[4]);
    }
    mysql_free_result(sql_result);
}
printf("\n\n");

```

mysql query

with trends as

```
(select brand_name, DATE_FORMAT(sell_date, '%%Y-%%m') as month, customer_id
from vehicle natural join model
where date_format(sell_date, '%%Y-%%m') >= date_format((NOW() - interval %d month), '%%Y-%%m')
order by brand_name, sell_date)
```

```
select brand_name, month, count(month) as sales,
count(case when gender = 'F' then 1 end) as Female,
count(case when gender = 'M' then 1 end) as Male
from trends natural join customer
group by brand_name, month
order by brand_name, month
```

TYPE 2-1-1 (TYPE 2-1을 수행하면 나오는 옵션이다.)

TYPE 2-1의 결과에 더해 소득별로 나누어준다. 이때도 역시 TYPE 1-1-1와 같은 방식으로 보여준다.

```
----- Subtypes in TYPE 2-1 -----
1. TYPE 2-1-1
0. QUIT
-----
Select Number : 1
----- TYPE 2-1-1 -----
** Then by income range. **
```

brand_name	month	sales	Female	Male	<F>0~1billion(₩)	<M>0~1billion(₩)	<F>1~2billion(₩)	<M>1~2billion(₩)	<F>2billion(₩)~	<M>2billion(₩)~
Audi	2020-06	1	0	1	0	1	0	0	0	0
Audi	2020-08	1	0	1	0	0	0	0	0	1
Audi	2020-09	1	0	1	0	0	0	0	0	1
Porsche	2020-08	1	1	0	0	0	0	1	0	0
Porsche	2021-03	1	1	0	0	0	1	0	0	0
Volkswagen	2020-11	1	0	1	0	0	0	1	0	0
Volkswagen	2021-03	1	0	1	0	0	0	0	0	1

code

```
printf("----- Subtypes in TYPE 2-1 -----\n");
printf("1. TYPE 2-1-1\n");
printf("0. QUIT\n");
printf("-----\n");
printf("Select Number : ");
scanf("%d", &subtype);
if (subtype != 1) break;

printf("\n----- TYPE 2-1-1 ----- \n\n");
printf("** Then by income range. **\n");
memset(query, 0, sizeof(query));
sprintf(query, "with trends as (select brand_name, DATE_FORMAT(sell_date, '%%Y-%%m') as month, customer_id from vehicle natural join model where date_format(sell_date, '%%Y-%%m') >= date_format((NOW() - interval %d month), '%%Y-%%m'))");
state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("brand_name month sales Female Male <F>0~1billion(₩) <M>0~1billion(₩) <F>1~2billion(₩) <M>1~2billion(₩) <F>2billion(₩)~ <M>2billion(₩)~\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-11s %-7s %-5s %-5s %-4s %-16s %-16s %-16s %-16s %-15s\n", sql_row[0], sql_row[1], sql_row[2], sql_row[3], sql_row[4], sql_row[5], sql_row[6], sql_row[7], sql_row[8], sql_row[9]);
    }
    mysql_free_result(sql_result);
}
```

mysql query

with trends as


```

(select brand_name, DATE_FORMAT(sell_date, '%Y-%m') as month, customer_id
from vehicle natural join model
where date_format(sell_date,'%Y-%m') >= date_format((NOW() - interval %d month),'%Y-%m')
order by brand_name, sell_date)

select brand_name, month, count(month) 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 annual_income >= 0 and annual_income < 100000000 and gender = 'F' then 1 end) as
'(F) 0~100000000',
count(case when annual_income >= 0 and annual_income < 100000000 and gender = 'M' then 1 end) as
'(M) 0~100000000',
count(case when annual_income >= 100000000 and annual_income < 200000000 and gender = 'F' then
1 end) as '(F) 0~200000000',
count(case when annual_income >= 100000000 and annual_income < 200000000 and gender = 'M' then
1 end) as '(M) 0~200000000',
count(case when annual_income >= 200000000 and gender = 'F' then 1 end) as '(F) 200000000~',
count(case when annual_income >= 200000000 and gender = 'M' then 1 end) as '(M) 200000000~'
from trends natural join customer
group by brand_name, month
order by brand_name, month

```

TYPE 3

date1~date2에 특정 회사에서 생산된 transmission이 결함이 있다고 가정한다.

date1과 date2와 supplier_name을 입력받아 date1과 date2 사이에 supplier의 공장에서 생성된 transmission을 찾아준다.

```

---- TYPE 3 ----
** Find that transmissions made by supplier (company name) between two given dates are defective. **
Which date1? : 20170301
Which date2? : 20200301
Which supplier? : Denso Corp.

component    supply_date  supplier_name
transmission  2018-03-26  Denso Corp.
transmission  2018-05-26  Denso Corp.
transmission  2018-08-26  Denso Corp.

```

code

```
printf("\n\n");
printf("---- TYPE 3 ----\n\n");
printf("++ Find that transmissions made by supplier (company name) between two given dates are defective. ++\n");
printf(" Which date1? : ");
fgets(date1, sizeof(date1), stdin);
date1[strlen(date1) - 1] = '\0';
printf(" Which date2? : ");
fgets(date2, sizeof(date2), stdin);
date2[strlen(date2) - 1] = '\0';
printf(" Which supplier? : ");
fgets(temp, sizeof(temp), stdin);
temp[strlen(temp) - 1] = '\0';
memset(query, 0, sizeof(query));
sprintf(query, "select component,supply_date, supplier_name from parts natural join supply_plant where component =

state = 0;
state = mysql_query(connection, query);
if (state == 0)
{
    printf("\ncomponent    supply_date    supplier_name\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-13s  %-11s  %-13s\n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}
printf("\n\n");
```

mysql query

```
select component,supply_date, supplier_name
from parts natural join supply_plant
where component = 'transmission' and
supply_date > %s and
supply_date < %s and
supplier_name = '%s'
```

TYPE 3-1 (TYPE 3을 수행하면 나오는 옵션이다.)

TYPE 3에서 구한 결함이 있는 transmission으로 만들어진 차를 구입한 고객을 찾아준다.

TYPE 3-2 (TYPE 3을 수행하면 나오는 옵션이다.)

TYPE 3에서 구한 결함이 있는 transmission으로 만들어진 차를 판매한 딜러를 찾아준다.

```
----- Subtypes in TYPE 3 -----
      1. TYPE 3-1
      2. TYPE 3-2
      0. QUIT
-----
Select Number : 2
----- TYPE 3-2 -----

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

VIN      dealer_id  dealer_name
1234578912345677  20160002  James
1234578912345678  20160002  James
1234578912345679  20160004  Lucas
```

code

```
else if (subtype == 2) {
    printf("----- TYPE 3-2 -----\n\n");
    printf("++ Find the dealer who sold the VIN and transmission for each vehicle containing these transmissions. ++\n");
    memset(query, 0, sizeof(query));
    sprintf(query, "with defective as (select S_ID, component, supply_date, supplier_name from parts natural join supply_p

    state = 0;
    state = mysql_query(connection, query);
    if (state == 0)
    {
        printf("\nVIN      dealer_id  dealer_name\n");
        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%-17s %-9s %-11s\n", sql_row[0], sql_row[1], sql_row[2]);
        }
        mysql_free_result(sql_result);
    }
}
```

mysql query

with defective as

(select S_ID, component, supply_date, supplier_name

from parts natural join supply_plant

where component = 'transmission' and

supply_date > %s and

supply_date < %s and

```
supplier_name = '%s')
```

```
select VIN, dealer_id, dealer_name  
from(defective natural join consist) natural join vehicle natural join dealer
```

TYPE 4

k를 입력받아 년도별 매출량 순위를 k등수 만큼 보여준다.

매출이 같은 경우 같은 등수로 표기 해주었다. K가 3일 때 1~3등을 보여주는데 해당연도에 팔린 브랜드가 2개 밖에 없다면 1~2등까지만 보여준다.

view를 만든 후 다 사용하고 출력이 끝나면 만든 view를 drop해주었다.

```
---- TYPE 4 ----  
  
** Find the top k brands by dollar-amount sold by the year. **  
Which K? : 3  
  
year  brand_name  rank  
2021  Porsche      1  
2021  Volkswagen    2  
2020  Audi           1  
2020  Porsche        2  
2020  Volkswagen      3  
2019  Audi           1  
2019  Porsche         2  
2019  Volkswagen      3
```

code

```

printf("\n\n");
printf("---- TYPE 4 ----\n\n");
printf("-- Find the top k brands by dollar-amount sold by the year. --\n");
printf(" Which K? : ");
scanf("%d", &k);
getchar();
memset(query, 0, sizeof(query));
sprintf(query, "create view dollar_amount as (select year(sell_date) as year, sum(price) as price, brand_name, customer_id from vehicle natural join model natural join total_price group by year(sell_date), brand_name having customer_id is not null);");
state = 0;
state = mysql_query(connection, query);

memset(query, 0, sizeof(query));
sprintf(query, "with dollar_amount_rank as (select year, price, brand_name, (@vRank := @Rank + 1), (@real_rank := IF ( @last > price, @real_rank:=@real_rank+1, @real_rank)), (@real_rank := IF(@prev_year < year,@real_rank:=1,@real_rank)) as real_rank, (@last := price), (@prev_year := year) from unit_sale, (select @vRank := 0,@real_rank :=1,@last:=0) as b order by year, price desc) select year, brand_name, real_rank as 'rank' from unit_sale_rank where real_rank <= 2;");
state = 0;
state = mysql_query(connection, query);

if (state == 0)
{
    printf("\nyear brandname rank\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-4s %-11s %-4s\n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
}

memset(query, 0, sizeof(query));
sprintf(query, "drop view if exists dollar_amount;");
state = 0;
state = mysql_query(connection, query);

```

mysql query

create view unit_sale as

```

(select year(sell_date) as year, sum(price)as price,brand_name, customer_id
from vehicle natural join model natural join total_price
group by year(sell_date),brand_name
having customer_id is not null);

```

with unit_sale_rank as

```

(select year,price,brand_name,
( @vRank := @Rank + 1 ),
( @real_rank := IF ( @last > price, @real_rank:=@real_rank+1, @real_rank)) ,
( @real_rank := IF(@prev_year < year,@real_rank:=1,@real_rank)) as real_rank,
( @last := price ),
( @prev_year := year )
from unit_sale, (select @vRank := 0,@real_rank :=1,@last:=0) as b
order by year, price desc)

```

```

select year, brand_name, real_rank as 'rank'
from unit_sale_rank
where real_rank <= 2;

```

```
drop view if exists unit_sale;
```

TYPE 5

k를 입력받아 년도별 판매량 순위를 k등수 만큼 보여준다.

매출이 같은 경우 같은 등수로 표기 해주었다. TYPE 4와 판매량인지 매출량인지 차이를 제외하면 같은 구조이다. 역시 view를 만든 후 다 사용하고 drop해주었다.

```
---- TYPE 5 ----  
  
** Find the top k brands by unit sales by the year. **  
Which K? : 2  
  
year  brand_name  rank  
2021  Porsche     1  
2021  Volkswagen   1  
2020  Audi          1  
2020  Volkswagen   2  
2019  Audi          1  
2019  Volkswagen   1  
2019  Porsche     2
```

code

```
printf("\n\n");  
printf("---- TYPE 5 ----\n\n");  
printf("** Find the top k brands by unit sales by the year. **\n");  
printf("Which K? : ");  
scanf("%d", &k);  
getchar();  
memset(query, 0, sizeof(query));  
sprintf(query, "create view unit_sale as (select year(sell_date) as year, count(sell_date) as cnt, brand_name, (@Rank:=0) as rank)");  
state = 0;  
state = mysql_query(connection, query);  
  
memset(query, 0, sizeof(query));  
sprintf(query, "with unit_sale_rank as (select year,cnt,brand_name, (@Rank:=0) as rank)");  
state = 0;  
state = mysql_query(connection, query);  
  
if (state == 0)  
{  
    printf("\nyear  brand_name  rank\n");  
    sql_result = mysql_store_result(connection);  
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)  
    {  
        printf("%-4s  %-11s  %-4s\n", sql_row[0], sql_row[1], sql_row[2]);  
    }  
    mysql_free_result(sql_result);  
}  
  
memset(query, 0, sizeof(query));  
sprintf(query, "drop view if exists unit_sale;");  
state = 0;  
state = mysql_query(connection, query);
```

mysql query

```
create view unit_sale as
(select year(sell_date) as year, count(brand_name)as cnt,brand_name, customer_id
from vehicle natural join model
group by year(sell_date),brand_name
having customer_id is not null);
```

```
with unit_sale_rank as
(select year,cnt,brand_name,
( @vRank := @Rank + 1 ),
( @real_rank := IF ( @last > cnt, @real_rank:=@real_rank+1, @real_rank)) ,
( @real_rank := IF(@prev_year < year,@real_rank:=1,@real_rank)) as real_rank,
( @last := cnt ),
( @prev_year := year )
from unit_sale, (select @vRank := 0,@real_rank :=1,@last:=0) as b
order by year, cnt desc)
```

```
select year, brand_name, real_rank as 'rank'
from unit_sale_rank
where real_rank <= 2;
```

```
drop view if exists unit_sale;
```

TYPE 6

convertible이 가장 많이 팔린 월을 찾아준다.

여기선 월 별이므로 년도는 나누어 주지 않고 년도에 상관없이 구해주었다.

```
---- TYPE 6 ----
** In what month(s) do convertibles sell best? **
month
8
```

code


```

printf("\n\n");
printf("---- TYPE 6 ----\n\n");
printf("++ In what month(s) do convertibles sell best? ++\n");
memset(query, 0, sizeof(query));
sprintf(query, "create view sale_convertible as (select sell_date from v
state = 0;
state = mysql_query(connection, query);

memset(query, 0, sizeof(query));
sprintf(query, "with month_sales as (select month(sell_date)as month, co
state = 0;
state = mysql_query(connection, query);

if (state == 0)
{
    printf("\nmonth\n");
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%-2s\n", sql_row[0]);
    }
    mysql_free_result(sql_result);
}

memset(query, 0, sizeof(query));
sprintf(query, "drop view if exists sale_convertible;");
state = 0;
state = mysql_query(connection, query);

```

mysql query

create view sale_convertible as

(select sell_date

from vehicle natural join model

where body_style = 'convertible' and sell_date is not null);

with month_sales as

(select month(sell_date)as month, count(month(sell_date)) as sales

from sale_convertible

group by month(sell_date)

order by month(sell_date))

select month

from month_sales

where sales = (select max(sales)

from month_sales);

drop view if exists sale_convertible;

TYPE 7

가장 오랫동안 재고를 가지고 있는 dealer를 찾아준다. 팔린 차는 sell_date에서 get_date를 빼주었고 아직 팔리지 않은 차는 오늘 날짜에서 get_date를 빼주어서 그 가장 차이가 큰 dealer의 이름을 구해주었다.

```
---- TYPE 7 ----  
  
** Find those dealers who keep a vehicle in inventory for the longest average time. **  
  
dealer_name  
Ethan
```

code

```
printf("\n\n");  
printf("---- TYPE 7 ----\n\n");  
printf("** Find those dealers who keep a vehicle in inventory for the longest average time. **\n");  
memset(query, 0, sizeof(query));  
sprintf(query, "with inventory as (select dealer_id,dealer_name, (case when sell_date is not null then  
state = 0;  
state = mysql_query(connection, query);  
if (state == 0)  
{  
    printf("\ndealer_name\n");  
    sql_result = mysql_store_result(connection);  
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)  
    {  
        printf("%-9s\n", sql_row[0]);  
    }  
    mysql_free_result(sql_result);  
}
```

mysql query

with inventory as

```
(select dealer_id, (case when sell_date is not null then DATEDIFF(sell_date,get_date) else  
DATEDIFF(NOW()),get_date)end) as date  
from vehicle natural join dealer)
```

select dealer_id

from inventory

where date = (select max(date) from inventory)