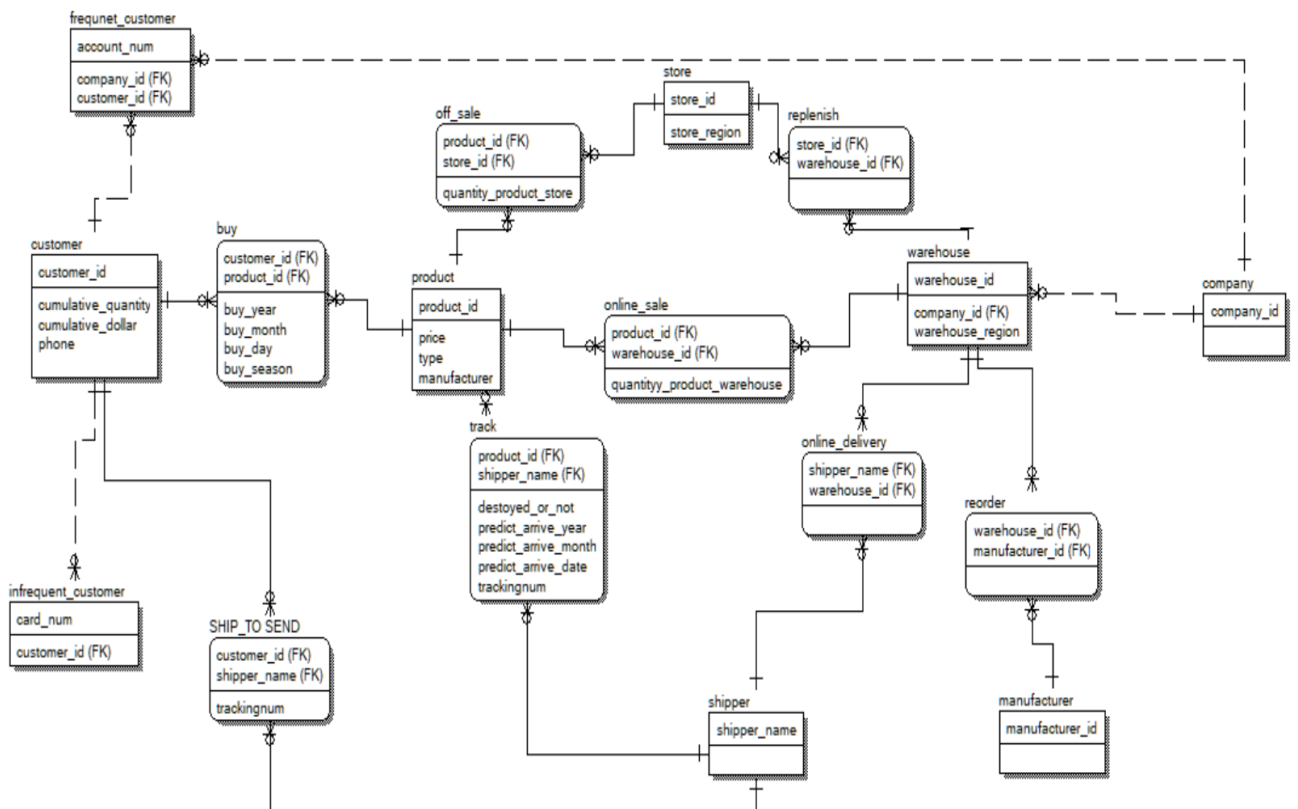


# 데이터베이스시스템 project 2 보고서

20190415 이규현

## 1. BCNF with logical schema diagram



1 . 목표 : Project1에서 작성한 Relational model에서 각 relation이 BCNF인지를 확인하는 과정을 보여준다.

warehouse(warehouse\_id, warehouse\_region,company\_id)

store에서 FD은 warehouse\_id→ warehouse\_region,company\_id만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

shipper(shipper\_id,)

shipper\_id에서 FD은 shipper\_id→shipper\_id만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Ship\_to\_send(customer\_id,shipper\_name,trackingnum)

Ship\_to\_send에서 FD은 customer\_id,shipper\_name →trackingnum 만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Infrequent\_customer(card\_num,customer\_id)

Infrequent\_customer에서 FD은 card\_num → customer\_id만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

frequent\_customer(account\_num,company\_id, customer\_id)

frequent\_customer에서 FD은 account\_num → customer\_id,company\_id만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

Reorder(warehouse\_id,manufacturer\_id)

reorder에서 FD은 warehouse\_id,manufacturer\_id → warehouse\_id,manufacturer\_id 만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Track(product\_id,shipper\_name, destroyed\_or\_not, predict\_arrive\_date\_trackingnum)

Track 에서 FD는

product\_id,shipper\_name →destroyed\_or\_not, predict\_arrive\_date\_trackingnum만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

company(company\_id,)

company에서 FD은 company\_id→company\_id만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

store(store\_id, store\_region)

store에서 FD은 store\_id→store region만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

manufacturer(manufacturer\_id,)

manufacturer 에서 FD은 manufacturer\_id→ manufacturer\_id만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

replenish(store\_id, warehouse\_id))

replenish에서 FD은 store\_id, warehouse\_id → store\_id, warehouse\_id `만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Online\_delivery(shipper\_name,warehouse\_id)

Online\_delivery에서 FD는 shipper\_name,warehouse\_id  $\rightarrow$  shipper\_name,warehouse\_id 만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Byu(customer\_id,product\_id,byu\_date,byu\_season)

buy 에서 FD은 customer\_id,product\_id  $\rightarrow$  byu\_date,byu\_season `만 존재한다.

이 FD는 자기자신으로 Trivial 하는 동시에 PK에 의한 종속이므로 BCNF이다

Off\_sale(product\_id, store\_id,quantity\_product\_store)

Off\_sale에서는 FD은 product\_id,store\_id $\rightarrow$ quantity\_product\_store만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

product(product\_id, price, type, manufacturer)

product에서 FD은 product\_id $\rightarrow$ price, type, manufacture만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

Customer(customer\_id, cumulative\_quantity cumulative dollar, phone)

customer에서 FD은 customer\_id $\rightarrow$ cumulative\_quantity, cumulative\_dollar, phone만 존재한다.

이 FD는 Trivial 하지 않으나 PK에 의한 종속이므로 BCNF이다

## 2.1 PHYSICAL SCHEMA

각각 테이블에 대해, 여러 도메인을 설정하고 설명한다.

### A . Customer

```
1 CREATE TABLE customer (  
2     customer_id VARCHAR(5) NOT NULL,  
3     cumulative_quantity VARCHAR(5),  
4     phone VARCHAR(15),  
5     cumulative_dollar VARCHAR(15),  
6     PRIMARY KEY (customer_id)  
7 );  
8
```

customer\_id: 회원의 아이디를 의미하고 문자열로 가정하였고, pk이므로 not null의 조건을 추가하였다.

cumulative\_quantity:: 회원의 구매수량을 의미한다.

cumulative dollar: 회원의 총 소비달러를 의미한다.

Phone : 회원의 정보인 핸드폰 번호에 해당한다.

### B. product

```
1 CREATE TABLE product (  
2     product_id VARCHAR(8) NOT NULL,  
3     price VARCHAR(10),  
4     ptype VARCHAR(10),  
5     manufacturer VARCHAR(10),  
6     PRIMARY KEY (product_id)  
7 );  
8  
9
```

product\_id: 물품의 고유번호를 의미하고 문자열로 가정하였다. pk이므로 not null의 조

건을 추가하였다.

Price : 물품의 구매가격을 의미한다.

Ptype: 물건의 타입을 의미한다.

manufacturer :물품의 제조사를 의미한다.

### C. COMPANY

```
1 CREATE TABLE company (  
2     company_id VARCHAR(10) NOT NULL,  
3     PRIMARY KEY (company_id)  
4 );  
5
```

COMPANY\_ID는 pk이므로 not null의 조건을 추가하였다.

### D. BUY

```
1 CREATE TABLE buy (  
2     customer_id VARCHAR(5) NOT NULL,  
3     product_id VARCHAR(8) NOT NULL,  
4     buy_season VARCHAR(3),  
5     buy_date DATE,  
6     PRIMARY KEY (product_id , customer_id),  
7     FOREIGN KEY (customer_id)  
8         REFERENCES customer (customer_id)  
9         ON DELETE CASCADE,  
10    FOREIGN KEY (product_id)  
11        REFERENCES product (product_id)  
12        ON DELETE CASCADE  
13 );  
14
```

CUSTOMER\_ID와 PRODUCT\_ID는 pk이므로 not null의 조건을 추가하였고 동시에 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다.

buy\_season과 BUY\_date 은 구매시기의 계절과 구매날짜를 의미한다..

## E. infrequent\_customer

```
Find  Find
1  CREATE TABLE infrequent_customer (
2      card_num VARCHAR(10) NOT NULL,
3      customer_id VARCHAR(5) NOT NULL,
4      PRIMARY KEY (card_num),
5      FOREIGN KEY (customer_id)
6          REFERENCES customer (customer_id)
7          ON DELETE CASCADE
8  );
9
```

Infrequent\_customer 은 자주 이용하지 않는 고객을 의미하고 이들은 카드번호로 식별 가능하기 때문에 pk이므로 not null의 조건을 추가하였다.

Customer\_id는 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

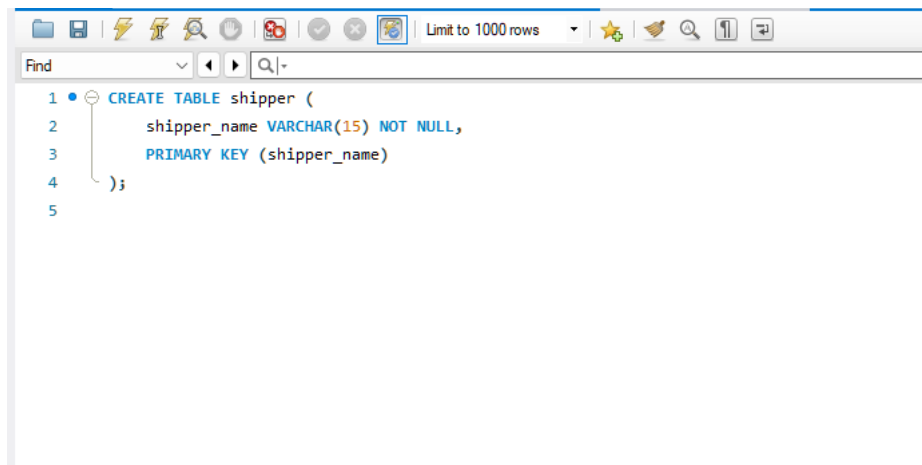
## F . frequent\_customer

```
Find  Find
1  CREATE TABLE frequent_customer (
2      account_num VARCHAR(12) NOT NULL,
3      company_id VARCHAR(10),
4      customer_id VARCHAR(5),
5      PRIMARY KEY (account_num),
6      FOREIGN KEY (customer_id)
7          REFERENCES customer (customer_id)
8          ON DELETE CASCADE,
9      FOREIGN KEY (company_id)
10         REFERENCES company (company_id)
11         ON DELETE CASCADE
12  );
13
```

frequent\_customer 은 자주 이용하는 고객을 의미하고 이들은 계좌번호로 식별 가능하기 때문에 pk이므로 not null의 조건을 추가하였다.

Customer\_id와 company\_id는 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

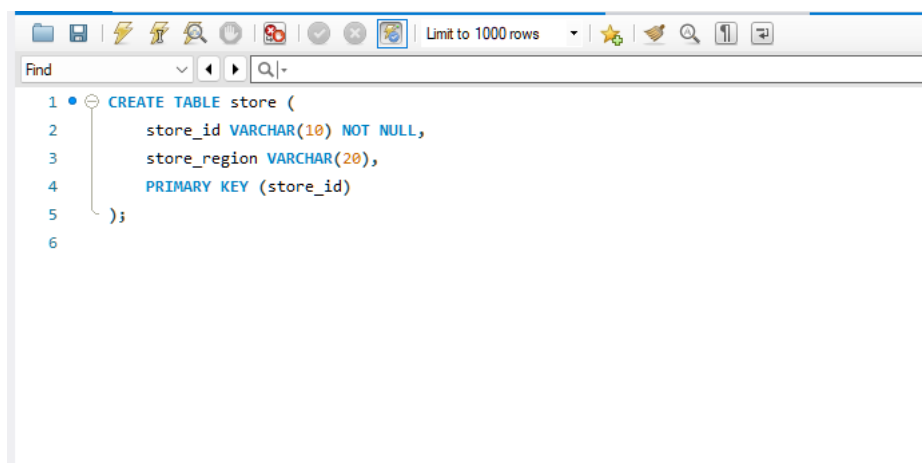
## G. shipper



```
1 CREATE TABLE shipper (  
2     shipper_name VARCHAR(15) NOT NULL,  
3     PRIMARY KEY (shipper_name)  
4 );  
5
```

Shipper\_name 은 운송회사 이름으로 pk이므로 not null의 조건을 추가하였다.

## H. store



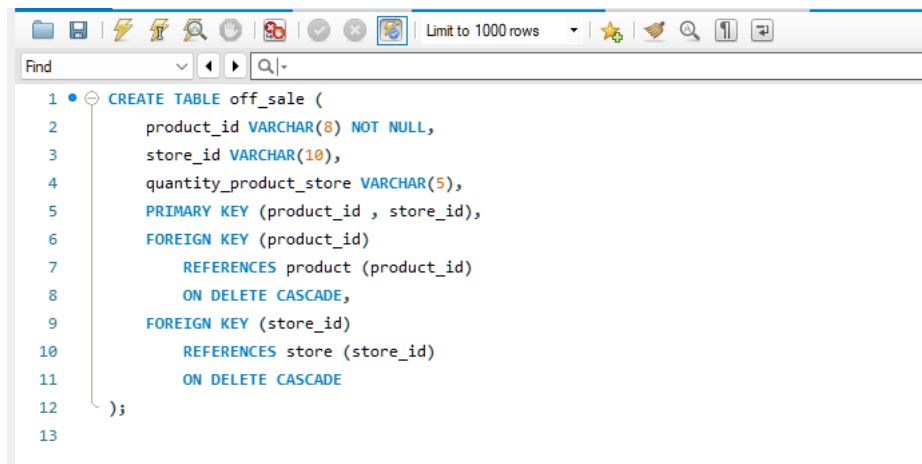
```
1 CREATE TABLE store (  
2     store_id VARCHAR(10) NOT NULL,  
3     store_region VARCHAR(20),  
4     PRIMARY KEY (store_id)  
5 );  
6
```

Store\_id 는 상점이름으로 pk이므로 not null의 조건을 추가하였다

Store\_region은 상점 지역을 나타내는 20개의 문자를 받을 수 있도록 하였다.



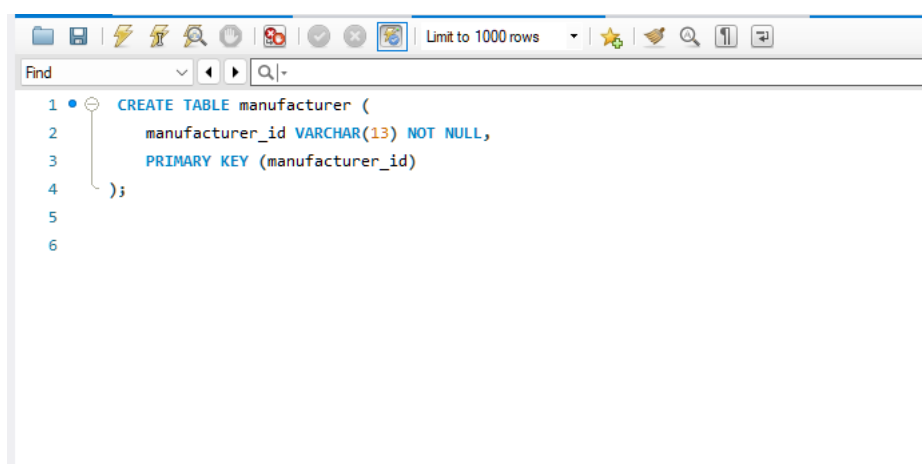
## I. Off\_sale



```
1 • CREATE TABLE off_sale (  
2     product_id VARCHAR(8) NOT NULL,  
3     store_id VARCHAR(10),  
4     quantity_product_store VARCHAR(5),  
5     PRIMARY KEY (product_id , store_id),  
6     FOREIGN KEY (product_id)  
7         REFERENCES product (product_id)  
8         ON DELETE CASCADE,  
9     FOREIGN KEY (store_id)  
10        REFERENCES store (store_id)  
11        ON DELETE CASCADE  
12 );  
13
```

Product\_id와 store\_id는 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

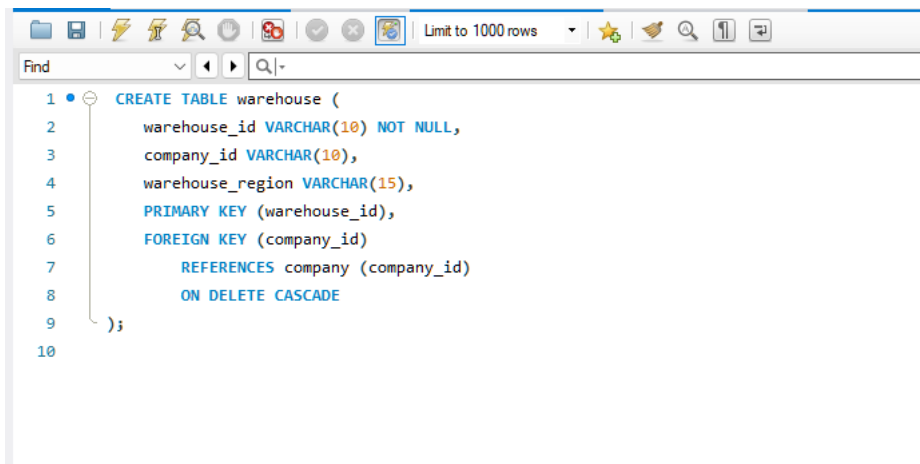
## J. manufacturer



```
1 • CREATE TABLE manufacturer (  
2     manufacturer_id VARCHAR(13) NOT NULL,  
3     PRIMARY KEY (manufacturer_id)  
4 );  
5  
6
```

Manufacturer\_id 는 pk이고 문자 13개를 받을 수 있도록 정하였다.

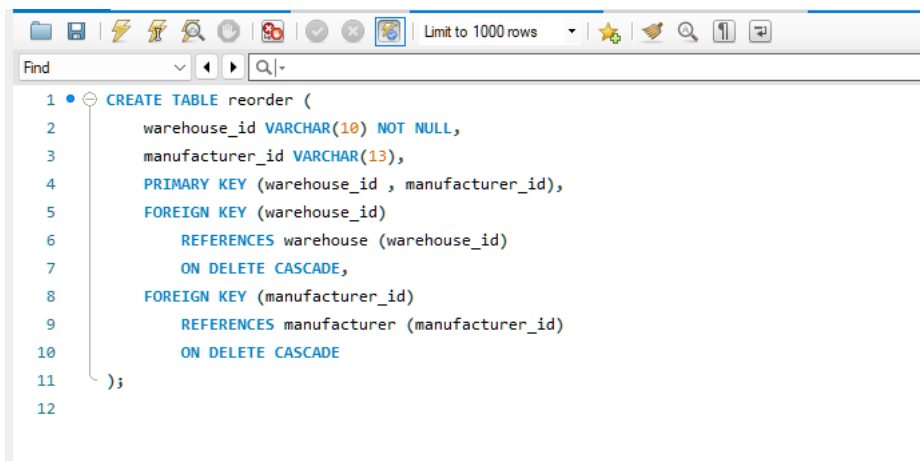
## K. warehouse



```
1 CREATE TABLE warehouse (  
2     warehouse_id VARCHAR(10) NOT NULL,  
3     company_id VARCHAR(10),  
4     warehouse_region VARCHAR(15),  
5     PRIMARY KEY (warehouse_id),  
6     FOREIGN KEY (company_id)  
7         REFERENCES company (company_id)  
8         ON DELETE CASCADE  
9 );  
10
```

Warehouse\_id 는 pk이므로 not null의 조건을 추가하였다. 그리고 company\_id는 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

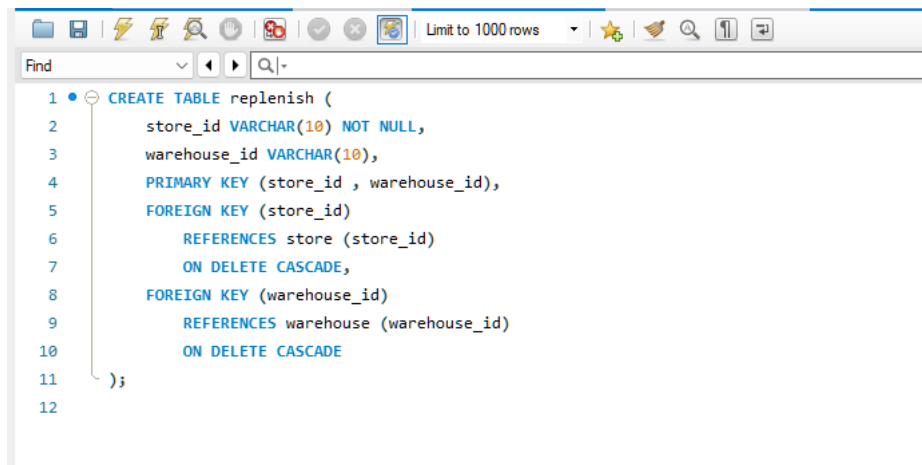
## L. reorder



```
1 CREATE TABLE reorder (  
2     warehouse_id VARCHAR(10) NOT NULL,  
3     manufacturer_id VARCHAR(13),  
4     PRIMARY KEY (warehouse_id , manufacturer_id),  
5     FOREIGN KEY (warehouse_id)  
6         REFERENCES warehouse (warehouse_id)  
7         ON DELETE CASCADE,  
8     FOREIGN KEY (manufacturer_id)  
9         REFERENCES manufacturer (manufacturer_id)  
10        ON DELETE CASCADE  
11 );  
12
```

Warehouse\_id와 manufacturer\_id가 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

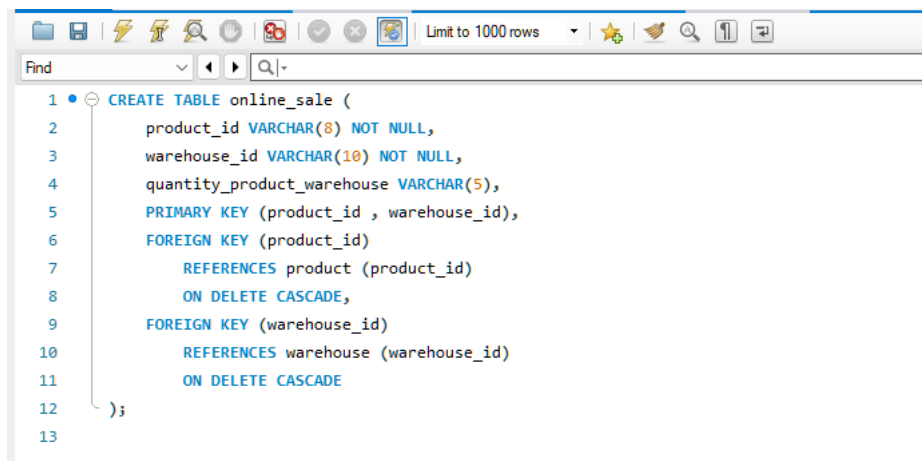
### m. replenish



```
1 CREATE TABLE replenish (  
2     store_id VARCHAR(10) NOT NULL,  
3     warehouse_id VARCHAR(10),  
4     PRIMARY KEY (store_id , warehouse_id),  
5     FOREIGN KEY (store_id)  
6         REFERENCES store (store_id)  
7         ON DELETE CASCADE,  
8     FOREIGN KEY (warehouse_id)  
9         REFERENCES warehouse (warehouse_id)  
10        ON DELETE CASCADE  
11 );  
12
```

Warehouse\_id와 store\_id가 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

### N. online\_sale

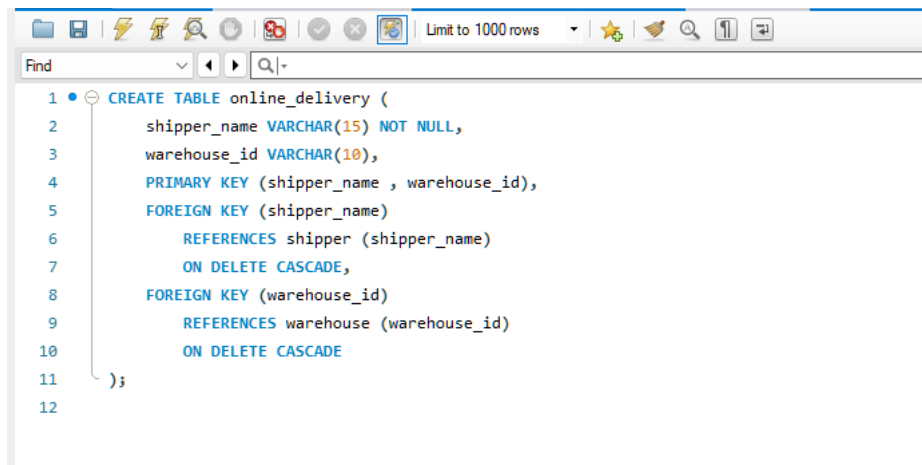


```
1 CREATE TABLE online_sale (  
2     product_id VARCHAR(8) NOT NULL,  
3     warehouse_id VARCHAR(10) NOT NULL,  
4     quantity_product_warehouse VARCHAR(5),  
5     PRIMARY KEY (product_id , warehouse_id),  
6     FOREIGN KEY (product_id)  
7         REFERENCES product (product_id)  
8         ON DELETE CASCADE,  
9     FOREIGN KEY (warehouse_id)  
10        REFERENCES warehouse (warehouse_id)  
11        ON DELETE CASCADE  
12 );  
13
```

Warehouse\_id와 product\_id가 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

Quantity\_product\_warehouse 는 창고의 재고량을 나타낸다.

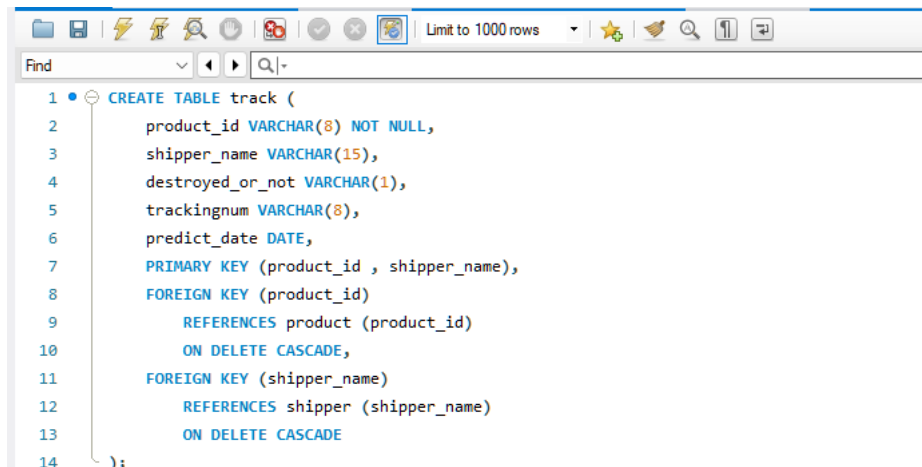
## O. Online\_delivery



```
1 CREATE TABLE online_delivery (  
2     shipper_name VARCHAR(15) NOT NULL,  
3     warehouse_id VARCHAR(10),  
4     PRIMARY KEY (shipper_name , warehouse_id),  
5     FOREIGN KEY (shipper_name)  
6         REFERENCES shipper (shipper_name)  
7         ON DELETE CASCADE,  
8     FOREIGN KEY (warehouse_id)  
9         REFERENCES warehouse (warehouse_id)  
10        ON DELETE CASCADE  
11 );  
12
```

Online\_delivery 는 Warehouse\_id와 shipper\_name이 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

## R. track

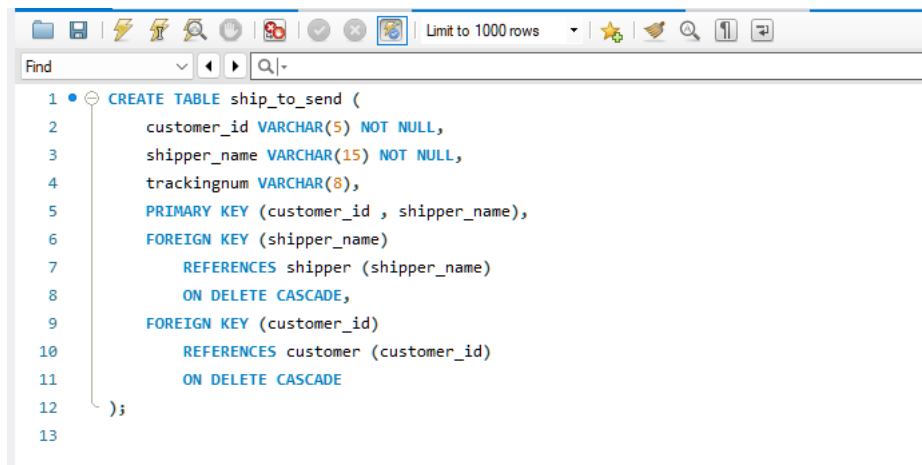


```
1 CREATE TABLE track (  
2     product_id VARCHAR(8) NOT NULL,  
3     shipper_name VARCHAR(15),  
4     destroyed_or_not VARCHAR(1),  
5     trackingnum VARCHAR(8),  
6     predict_date DATE,  
7     PRIMARY KEY (product_id , shipper_name),  
8     FOREIGN KEY (product_id)  
9         REFERENCES product (product_id)  
10        ON DELETE CASCADE,  
11     FOREIGN KEY (shipper_name)  
12        REFERENCES shipper (shipper_name)  
13        ON DELETE CASCADE  
14 );
```

Track은 product id와 shipper\_name이 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다

Destroyed or not은 파손여부 , trackingnum은 추적번호를 문자열로 여유있게 구성하였다.

## S. ship\_to\_send

A screenshot of a SQL IDE window. The title bar includes a toolbar with icons for file operations, a search bar, and a 'Limit to 1000 rows' dropdown. The main text area displays a SQL script for creating a table named 'ship\_to\_send'. The script is numbered 1 through 13 on the left margin. The SQL code defines columns 'customer\_id' (VARCHAR(5) NOT NULL), 'shipper\_name' (VARCHAR(15) NOT NULL), and 'trackingnum' (VARCHAR(8)). It sets a primary key on 'customer\_id' and 'shipper\_name', and two foreign keys: one for 'shipper\_name' referencing the 'shipper' table, and another for 'customer\_id' referencing the 'customer' table. Both foreign keys have 'ON DELETE CASCADE' behavior. The script ends with a semicolon on line 12.

```
1 CREATE TABLE ship_to_send (  
2     customer_id VARCHAR(5) NOT NULL,  
3     shipper_name VARCHAR(15) NOT NULL,  
4     trackingnum VARCHAR(8),  
5     PRIMARY KEY (customer_id , shipper_name),  
6     FOREIGN KEY (shipper_name)  
7         REFERENCES shipper (shipper_name)  
8         ON DELETE CASCADE,  
9     FOREIGN KEY (customer_id)  
10        REFERENCES customer (customer_id)  
11        ON DELETE CASCADE  
12 );  
13
```

ship\_to\_send는 customer\_id와 shipper\_name이 pk이므로 not null의 조건을 추가하였다. 동시에 둘 다 FK이므로 참조제약을 어기지 않도록 ON DELETE CASCADE를 붙여주었다 , trackingnum은 추적번호를 문자열로 여유있게 8 문자로 지정하였다.

### 3. MY SQL, OBCD C LANGUAGE

1. TYPE 1 을 위한 쿼리를 작성하였다.



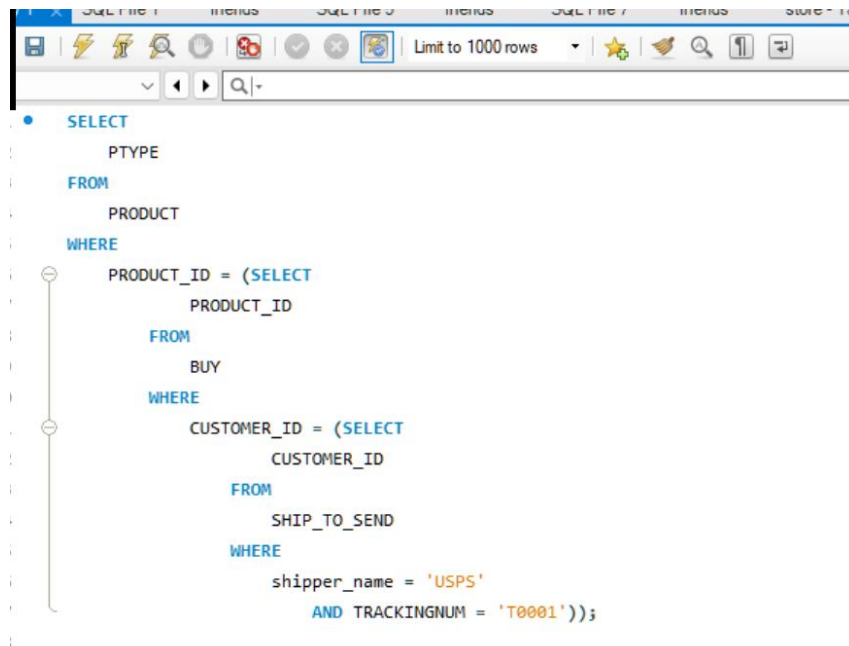
실제 VS에 입력한 코드는

```
sprintf(query, "SELECT PHONE FROM CUSTOMER WHERE CUSTOMER_ID = (SELECT  
CUSTOMER_ID FROM SHIP_TO_SEND WHERE shipper_name = '%s' AND TRACKINGNUM  
= '%s');", "USPS", tracknum);
```

으로 TRACKNUM을 입력 받을 수 있도록 하였다.

Tracknumdms T0001 – T0015까지 입력이 가능하다.

Type 1-1은 1 을 누르면 결과가 나온다.



실제 입력 쿼리는 이러하다.

```

sprintf(query2, "SELECT PTYPE FROM PRODUCT WHERE PRODUCT_ID = (SELECT
PRODUCT_ID FROM BUY WHERE CUSTOMER_ID = (SELECT CUSTOMER_ID FROM
SHIP_TO_SEND WHERE shipper_name = '%s' AND TRACKINGNUM = '%s'))";, "USPS",
tracknum);

```

'usps'에서 추적번호 'T0001'로 오는 고객을 구매내역을 담당하는 buy에서 어떤 물건을 샀는지 확인하고 그것이 어떤 타입의 물건인지 확인하였다.

그리고 새로운 배송을 요청하는 쿼리는 다음과 같다.

```

sprintf(query3, " SET @VAR1 = (SELECT CUSTOMER_ID FROM SHIP_TO_SEND WHERE
shipper_name = '%s' AND TRACKINGNUM = '%s');\n

```

```

DELETE FROM SHIP_TO_SEND WHERE CUSTOMER_ID =
@VAR1 AND SHIPPER_NAME = '%s';\n

```

```

INSERT INTO SHIP_TO_SEND VALUES(@VAR1, '%s', '%s');, "USPS",
tracknum,"USPS","USPS",tracknum);

```

```

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 -----
** Assume the package shipped by USPS with tracking number X is reported to have been destroyed in an accident. Find the contact information for the customer. **
Which X? : T0001

-----
shipper : USPS   tracking num : T0001
-----
CUSTOMER_ID : 0001   PHONE NUMBER : 010-8498-9877

----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. RESTART TYPE 1
0. QUIT TO SELECT QUERY TYPES
-----
1
----- TYPE 1-1 -----
**Then find the contents of that shipment and create a new shipment of replacement items. **
PRODUCT TYPE : A

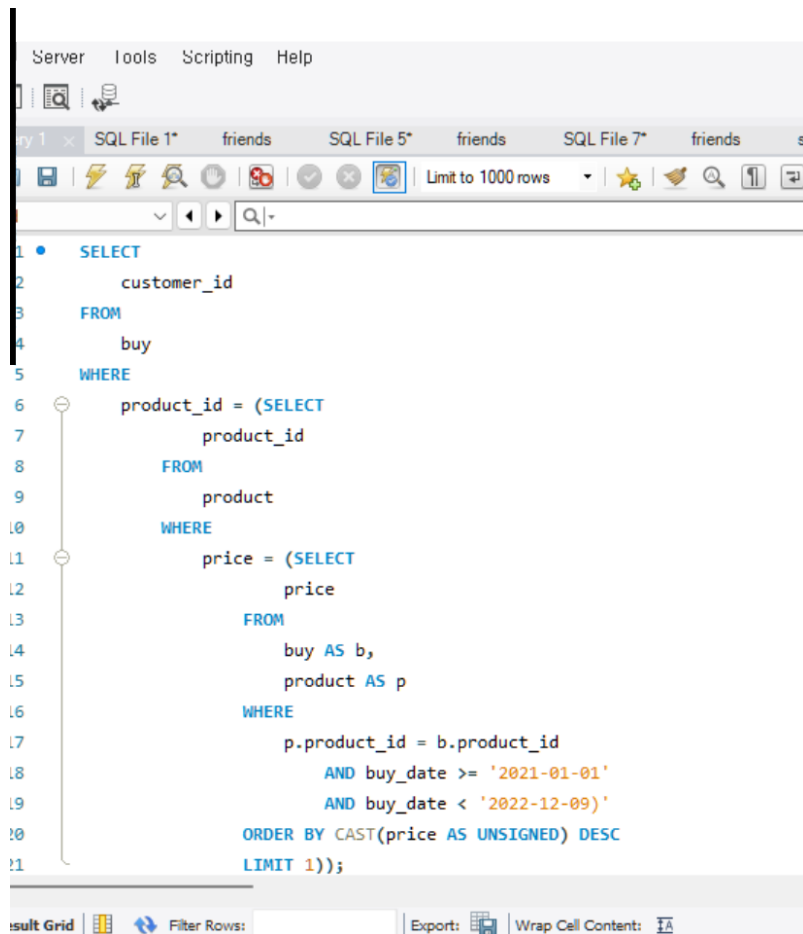
----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. RESTART TYPE 1
0. QUIT TO SELECT QUERY TYPES
-----

```

다음과 같이 USPS의 T0001 추적 번호의 물건을 시키는 손님은 0001이라는 손님이고 그 손님이 산 물건의 타입은 A라는 타입의 물건이다.



2. Type2를 해결하기 위한 쿼리는 이렇다.,



실제 입력 쿼리는 이렇다.

```
sprintf(query, "select customer_id from buy where product_id = ₩
```

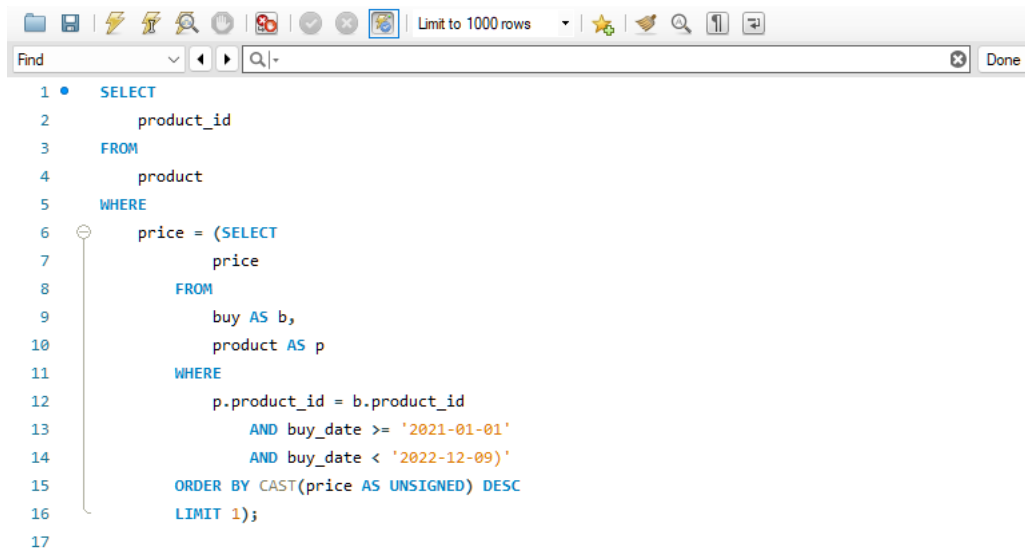
```
(select product_id from product where price = (select price from ₩
```

```
buy as b,product as p where p.product_id = b.product_id and buy_date ₩
```

```
>='2021-01-01' and buy_date<'2022-12-09' order by cast(price as  
unsigned) desc limit 1));");
```

작년을 기준으로 날짜값을 고정하고 BUY와 PRODUCT를 조인하여 가장 많이 팔린 물건을 찾고 구매내역을 나타내는 BUY를 통해 그것을 산 사람을 찾아내었다.

Type 2-1 의 쿼리는 다음과 같다.



```
1 • SELECT
2     product_id
3 FROM
4     product
5 WHERE
6     price = (SELECT
7         price
8     FROM
9         buy AS b,
10        product AS p
11    WHERE
12        p.product_id = b.product_id
13        AND buy_date >= '2021-01-01'
14        AND buy_date < '2022-12-09'
15    ORDER BY CAST(price AS UNSIGNED) DESC
16    LIMIT 1);
17
```

실제 입력쿼리는 이러하다.

```
sprintf(query2, "select product_id from product where price = (select price from product
    join buy as b, product as p where p.product_id =
    b.product_id and buy_date >= '2021-01-01' and buy_date < '2022-12-09'
    order by cast(price as unsigned) desc limit 1);");
```

작년을 기준으로 고정하고 buy와 product의 join을 통해 가장 많이 팔린 것을 찾아내고  
그 물품의 이름을 찾아내었다.

```
강대학교
GANG UNIVERSITY
Visual C++... HOfficeDN... Irvine 7th E... 곰플레이어 Scan4.pdf 사실확인서... TeamViewer HW_2019

C:\Users\Wgyu_Hyun\source\repos\Project55\Wx64\Debug\Project55.exe

----- 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 -----
** Find the customer who has bought the most (by price) in the past year **
CUSTOMER_ID : 0006
CUSTOMER_ID : 0011
CUSTOMER_ID : 0015

----- Subtypes in TYPE2 -----
1. TYPE 2-1
2. RESTART TYPE 2
0. QUIT TO SELECT QUERY TYPES

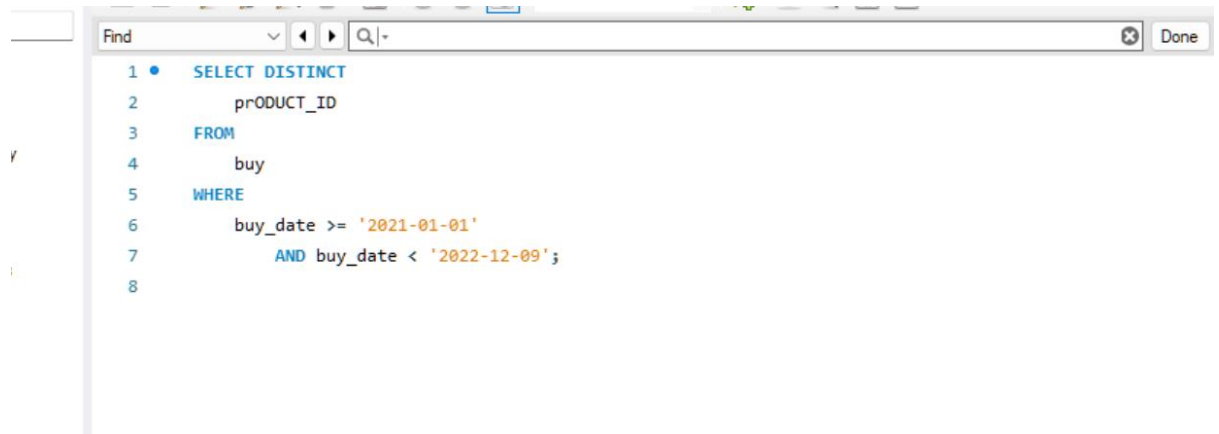
1

----- TYPE 2-1 -----
** ) Then find the product that the customer bought the most. (**
PRODUCT_ID : PD07

----- Subtypes in TYPE2 -----
1. TYPE 2-1
2. RESTART TYPE 2
0. QUIT TO SELECT QUERY TYPES
```

작년에 가장 많이 돈을 쓴 사람을 찾고 가장 많이 팔린 제품을 찾았다.

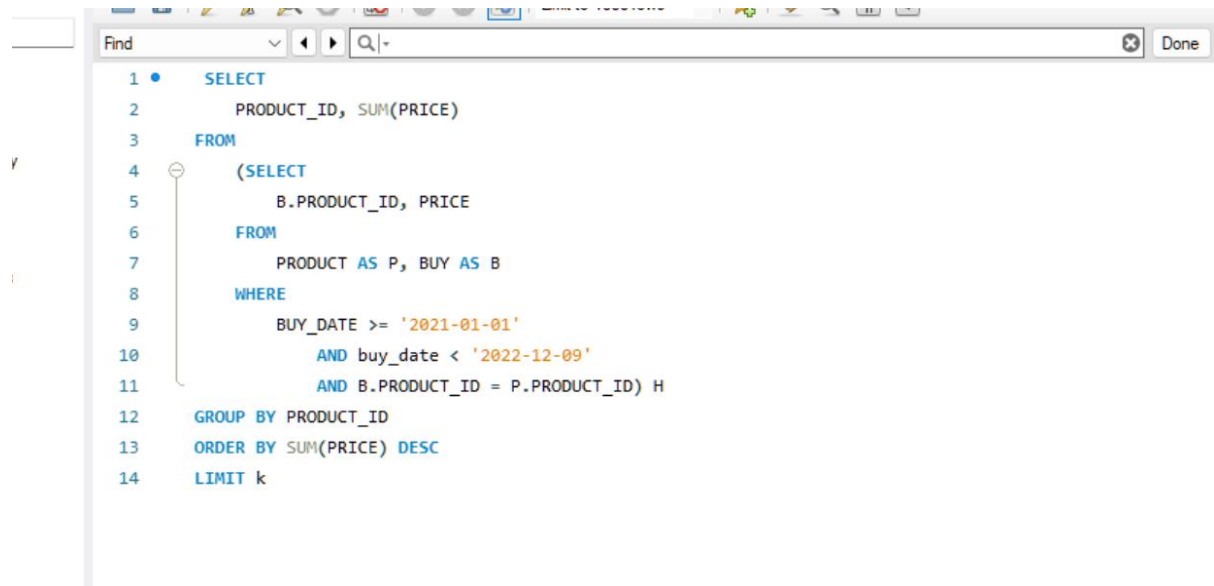
### 3. Type 3



실제 입력 쿼리 `sprintf(query, "select DISTINCT prODUCT_ID from buy whereW`  
`buy_date >='2021-01-01' and buy_date<'2022-12-09';W`  
`");`

작년을 기준으로 팔린 물품 품목을 나열하였다.

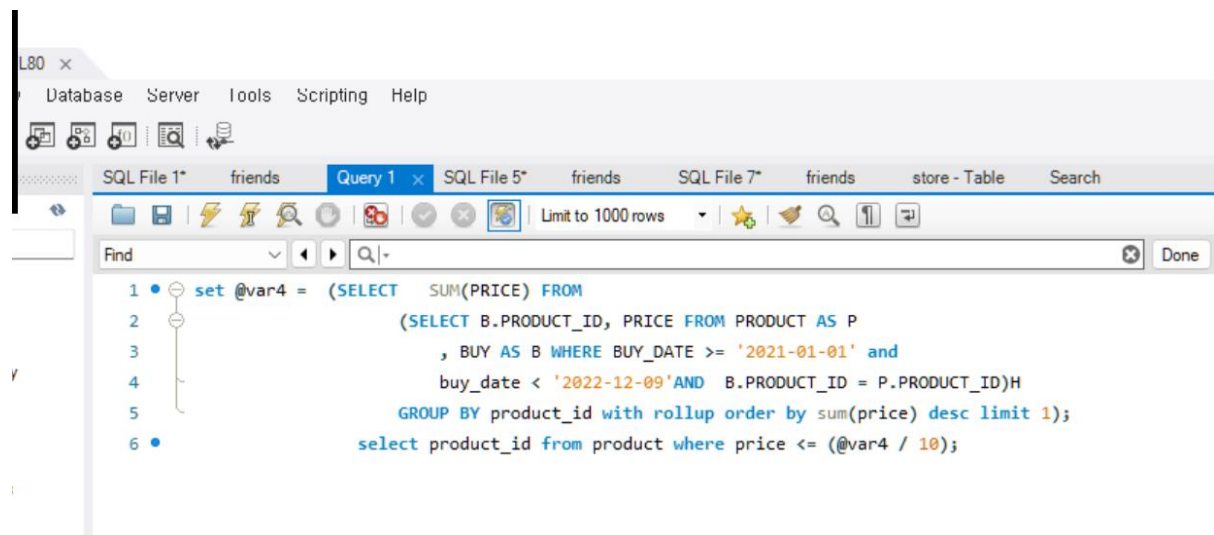
### Type 3-1



```
1 • SELECT
2     PRODUCT_ID, SUM(PRICE)
3 FROM
4     (SELECT
5         B.PRODUCT_ID, PRICE
6     FROM
7         PRODUCT AS P, BUY AS B
8     WHERE
9         BUY_DATE >= '2021-01-01'
10        AND buy_date < '2022-12-09'
11        AND B.PRODUCT_ID = P.PRODUCT_ID) H
12 GROUP BY PRODUCT_ID
13 ORDER BY SUM(PRICE) DESC
14 LIMIT k
```

작년 기준으로 상위 k번째로 많이 매출을 올린 것을 찾기 위해 product와 buy를 join하여 그 물품과 판매 매출을 뽑아내었다.

### Type 3-2



```
1 • set @var4 = (SELECT SUM(PRICE) FROM
2     (SELECT B.PRODUCT_ID, PRICE FROM PRODUCT AS P
3     , BUY AS B WHERE BUY_DATE >= '2021-01-01' and
4     buy_date < '2022-12-09' AND B.PRODUCT_ID = P.PRODUCT_ID)H
5     GROUP BY product_id with rollup order by sum(price) desc limit 1);
6 • select product_id from product where price <= (@var4 / 10);
```

워크 벤치에서는 작동하지만 vs에서는 버전차이 때문에 인식이 안되는 듯 하다.

변수 설정을 통해 그 변수를 사용하여 매출의 상위 10퍼 제품을 찾아내었다. Select의 결과 테이블을 테이블로서 활용하였다.

```
C:\Users\Gyu_Hyun\source\repos\Project55\64\Debug\Project55.exe

----- 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
-----
3
QUERY TYPES: 3

----- TYPE 3 -----
** Find ALL products sold in the past year **
product_id : PD01
product_id : PD02
product_id : PD03
product_id : PD04
product_id : PD05
product_id : PD07

----- Subtypes in TYPE3 -----
1. TYPE 3-1
2. TYPE 3-2
3. RESTART TYPE 3
0. QUIT TO SELECT QUERY TYPES
-----
1

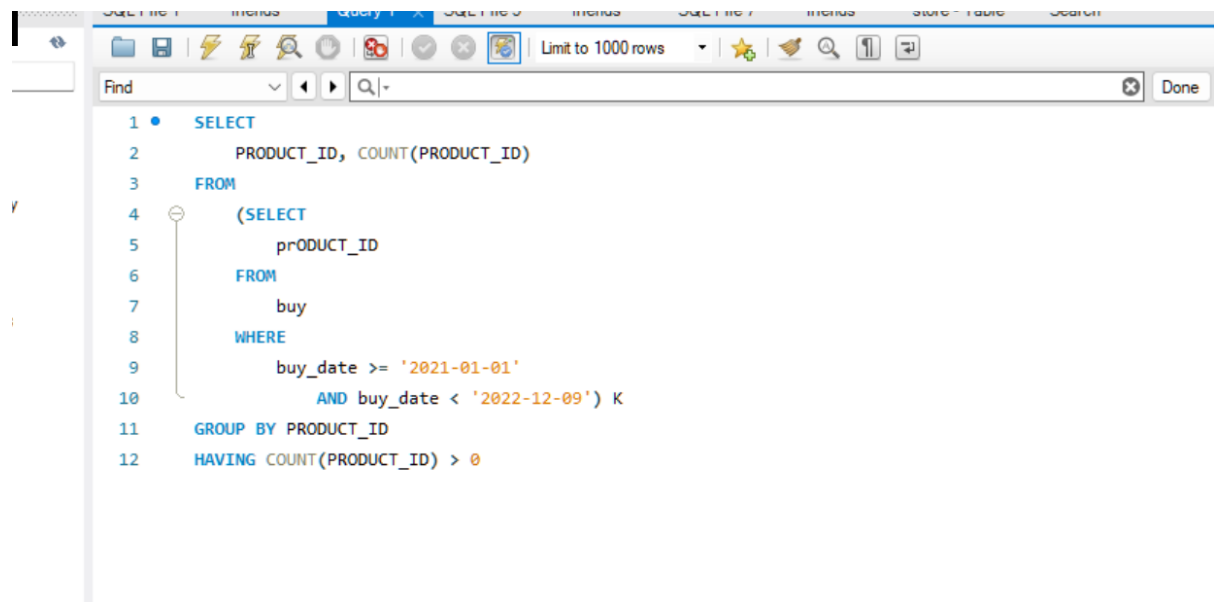
----- TYPE 3-1 -----
** ) Then find the top k products by dollar-amount sold. **
WHAT IS K?1
product_id, sum(price) : PD07    30000

----- Subtypes in TYPE3 -----
1. TYPE 3-1
2. TYPE 3-2
3. RESTART TYPE 3
0. QUIT TO SELECT QUERY TYPES
-----
```

작년에 팔린 제품을 모두 뽑아내고

달러 소비량 중에 top k 를 입력받아 구현하였고 그 물건과 팔린 가격도 뽑아내었다.

Type 4



```
1 • SELECT
2   PRODUCT_ID, COUNT(PRODUCT_ID)
3 FROM
4   (SELECT
5     product_ID
6   FROM
7     buy
8   WHERE
9     buy_date >= '2021-01-01'
10    AND buy_date < '2022-12-09') K
11 GROUP BY PRODUCT_ID
12 HAVING COUNT(PRODUCT_ID) > 0
```

작년 기준 팔린 모든 제품을 뽑아내었다. Having 절과 count를 결합하여 활용하였다.

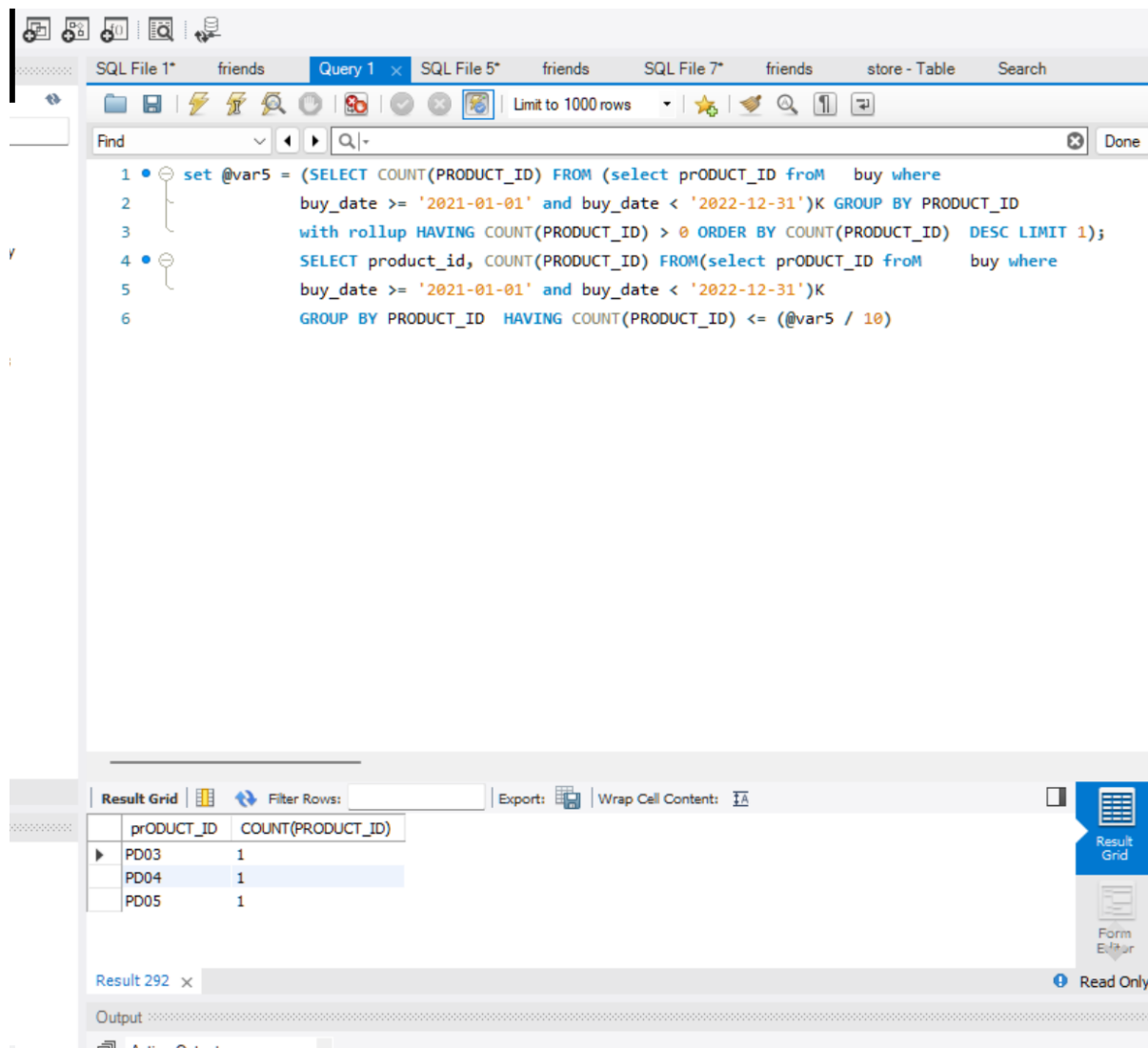
Type 4-1



```
1 • SELECT
2   PRODUCT_ID, COUNT(PRODUCT_ID)
3 FROM
4   (SELECT
5     product_ID
6   FROM
7     buy
8   WHERE
9     buy_date >= '2021-01-01'
10    AND buy_date < '2022-12-09') K
11 GROUP BY PRODUCT_ID
12 HAVING COUNT(PRODUCT_ID) > 0
13 ORDER BY COUNT(PRODUCT_ID) DESC
14 LIMIT k
```

그 중에서 상위 k개를 뽑아내기 위해 limit을 사용하고 제품과 판매량을 뽑아내었다.

#### Type 4-2



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is as follows:

```
1 • set @var5 = (SELECT COUNT(PRODUCT_ID) FROM (select prODUCT_ID frOm buy where
2 •      buy_date >= '2021-01-01' and buy_date < '2022-12-31')K GROUP BY PRODUCT_ID
3 •      with rollup HAVING COUNT(PRODUCT_ID) > 0 ORDER BY COUNT(PRODUCT_ID) DESC LIMIT 1);
4 •
5 •      SELECT product_id, COUNT(PRODUCT_ID) FROM(select prODUCT_ID frOm buy where
6 •      buy_date >= '2021-01-01' and buy_date < '2022-12-31')K
       GROUP BY PRODUCT_ID HAVING COUNT(PRODUCT_ID) <= (@var5 / 10)
```

The result grid shows the following data:

prODUCT_ID	COUNT(PRODUCT_ID)
PD03	1
PD04	1
PD05	1

The result grid is titled "Result 292" and has a "Read Only" button. The output pane at the bottom is empty.

이것또한 변수 설정을 통해 워크 벤치에서는 작동하지만 vs에서는 버전차이 때문에 인식이 안되는 듯 하다.

변수 설정을 통해 그 변수를 사용하여 판매량의 상위 10퍼 제품을 찾아내었다. Select의 결과 테이블을 테이블로서 활용하였다.



## Type 4 실행 결과

```
C:\Users\Gyu_Hyun\source\repos\Project55\64\Debug\Project55.exe
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
-----
4
QUERY TYPES: 4

----- TYPE 4 -----
** Find all products by unit sales in the past year**

-----
PRODUCT_ID AND COUNT : PD01 2
PRODUCT_ID AND COUNT : PD02 2
PRODUCT_ID AND COUNT : PD03 1
PRODUCT_ID AND COUNT : PD04 1
PRODUCT_ID AND COUNT : PD05 1
PRODUCT_ID AND COUNT : PD07 3

----- Subtypes in TYPE4 -----
1. TYPE 4-1
2. TYPE 4-2
3. RESTART TYPE 3
0. QUIT TO SELECT QUERY TYPES
-----
1

----- TYPE 4-1 -----
** Then find the top k products by unit sales.. **

-----
WHAT IS K?1
-----
product_id and count PD07 3
```

작년에 팔린 제품과 개수를 뽑아냈고, 상위 k개의 개수 판매량을 기록하는 제품과 소비  
갯수를 뽑아내었다.

## Type 5

```
1  SELECT
2      PRODUCT_ID
3  FROM
4      OFF_SALE AS A,
5      STORE AS B
6  WHERE
7      A.STORE_ID = B.STORE_ID
8      AND STORE_REGION = 'CALI'
9      AND QUANTITY_PRODUCT_STORE = '0';
10
```

캘리포니아에 재고가 없는 물품을 stoer 와 오프라인 고객이 사는 행위를 나타내는 off\_sale을 통해 찾아내었다.

```

----- 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
-----
5
QUERY TYPES: 5

----- TYPE 5 -----

** Find those products that are out-of-stock at every store in California.**

-----

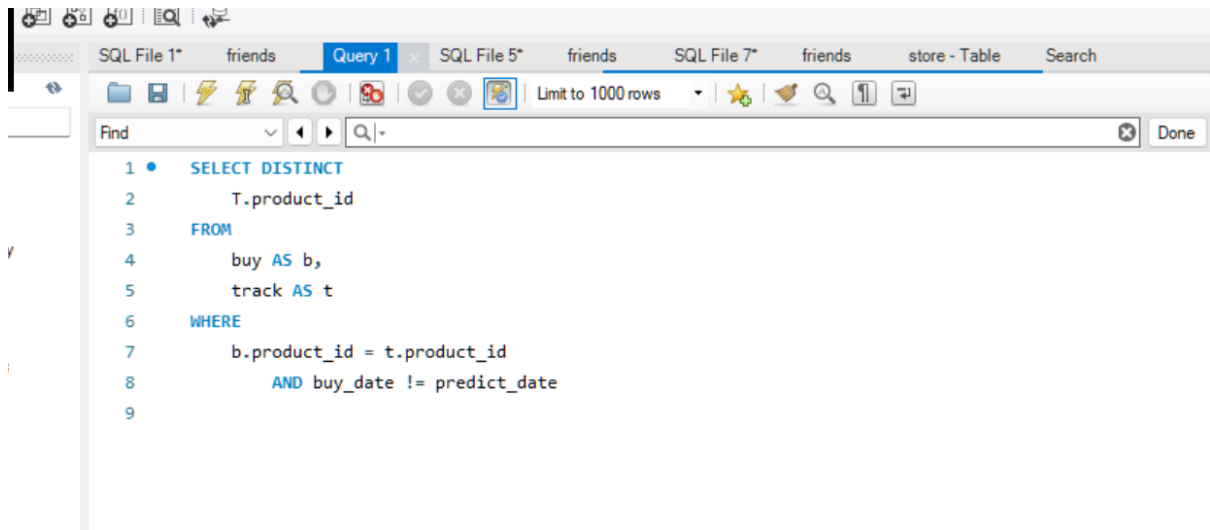
-----
SOLD OUT IN CALI STORE : PRODUCT_ID : PD04
SOLD OUT IN CALI STORE : PRODUCT_ID : PD05

-----

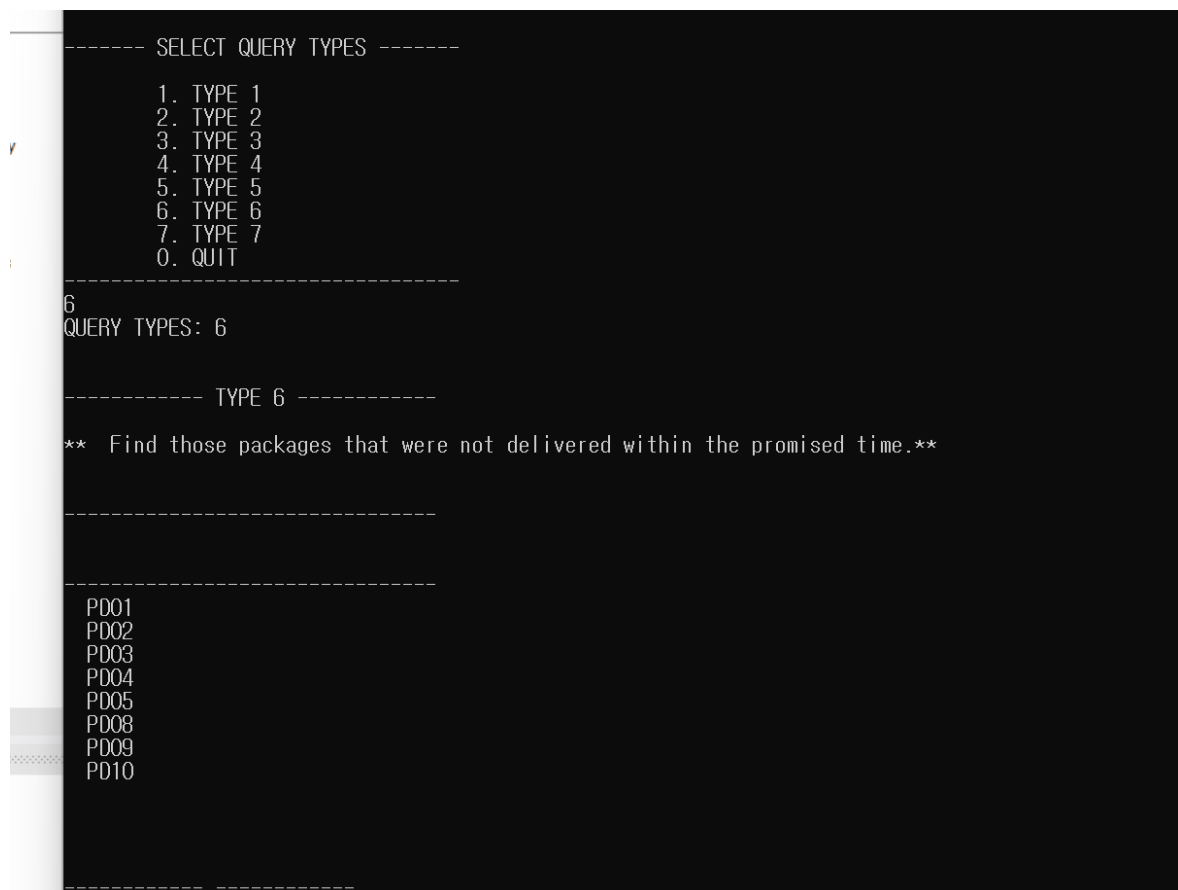
1. RESTART TYPE 1
0. QUIT TO SELECT QUERY TYPES
-----

```

## Type 6



약속된 시간에 도착하지 않은 물건을 buy와 track을 join한 결과로 찾아내어 뽑아내었다. 실제 배송날과 예상일의 차이가 있는 것을 찾아 뽑아내었다. 아래는 실행 결과이다.



## Type 7

```
1 • SELECT
2     B.CUSTOMER_ID, P.PRODUCT_ID, P.PRICE, BUY_DATE
3 FROM
4     BUY AS B,
5     PRODUCT AS P
6 WHERE
7     B.PRODUCT_ID = P.PRODUCT_ID
8     AND BUY_DATE > '2022-05-01'
9     AND BUY_DATE < '2022-06-01'
10
```

기간을 고정하고 buy와 product 를 join하여 지난 달의 고객들이 구매한 내역을 뽑아내었다. 아래는 출력 결과이다.

```
----- 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

7
QUERY TYPES: 7

----- TYPE 7 -----
**Generate the bill for each customer for the past month. .**

-----
0012 PD04 3000 2022-05-13
0006 PD07 10000 2022-05-03
0011 PD07 10000 2022-05-14

-----

1. RESTART TYPE 1
0. QUIT TO SELECT QUERY TYPES
-----
```



