

# **CSE4110 Database System**

DB Project 2

20181294 임승섭

## 1. 기존 모델에서 수정 사항

1. bill과 package의 관계를 one to many 관계(many에 total)로 변경한다

여러 개의 택배를 한 번에 결제할 수 있기 때문에 bill 이 one side가 되고, package가 many side가 된다. 따라서 이를 reduction한 결과도 수정한다. bill의 primary key(bill\_ID)를 package의 attribute로 추가해주었다.

bill (bill\_ID, payer\_ID, method, price, billing\_date, payment\_date, bill\_status)

package(package\_ID, shipper\_ID, box\_type, weight, delivery type, delivery status, start day, end day, promised day, special note, customer\_ID, company\_name, bill\_ID)

## 2. Relation의 functional dependency

1. shipper (shipper\_ID, name, address, account\_num)

shipper\_ID -> name, address, account\_num

2. customer (customer\_ID, name, address, account\_num)

customer\_ID -> name, address, account\_num

3. contract (shipper\_ID, customer\_ID)

trivial

4. user-shipper (company\_name, shipper\_ID, num\_of\_use, total\_payment\_cost)

company\_name, shipper\_ID -> num of use, total payment cost

5. user-customer (company\_name, customer\_ID, num\_of\_use, total\_payment\_cost)

company\_name, shipper\_ID -> num of use, total payment cost

6. shipping company (**company\_name**, contact\_number)

company\_name -> contact number

7. package (**package\_ID**, shipper\_ID, customer\_ID, company\_name, box\_type, weight, delivery\_type, deliver\_status, start\_day, end\_day, promised\_day, special\_note, bill\_ID)

package\_ID -> shipper\_ID, customer\_ID, company\_name, box\_type, weight, delivery\_type, delivery\_status, start\_day, end\_day, promised\_day, special\_note, bill\_ID

8. bill (**bill\_ID**, payer\_ID, method, price, billing\_date, payment\_date, bill\_status)

bill\_ID -> payer\_ID, method, price, billing\_date, payment\_date, bill\_status

9. deliver (**package\_ID**, vehicle\_ID, vehicle\_type)

package\_ID -> vehicle\_ID, vehicle\_type

10. vehicle (**vehicle\_ID, vehicle\_type**, licesne\_ID, maximum\_loads, vehicle\_status)

vehicle\_ID, vehicle\_type -> licesne\_ID, maximum\_loads, vehicle\_status

11. driver (**license\_ID**, name, phone\_number)

license\_ID -> name, phone\_number

12. start\_point (**vehicle\_ID, vehicle\_type**, warehouse\_address)

vehicle\_ID, vehicle\_type -> warehouse\_address

13. destination (**vehicle\_ID, vehicle\_type**, warehouse\_address)

vehicle\_ID, vehicle\_type -> warehouse\_address

14. warehouse (warehouse\_address, warehouse\_name)

warehouse\_address -> warehouse\_name

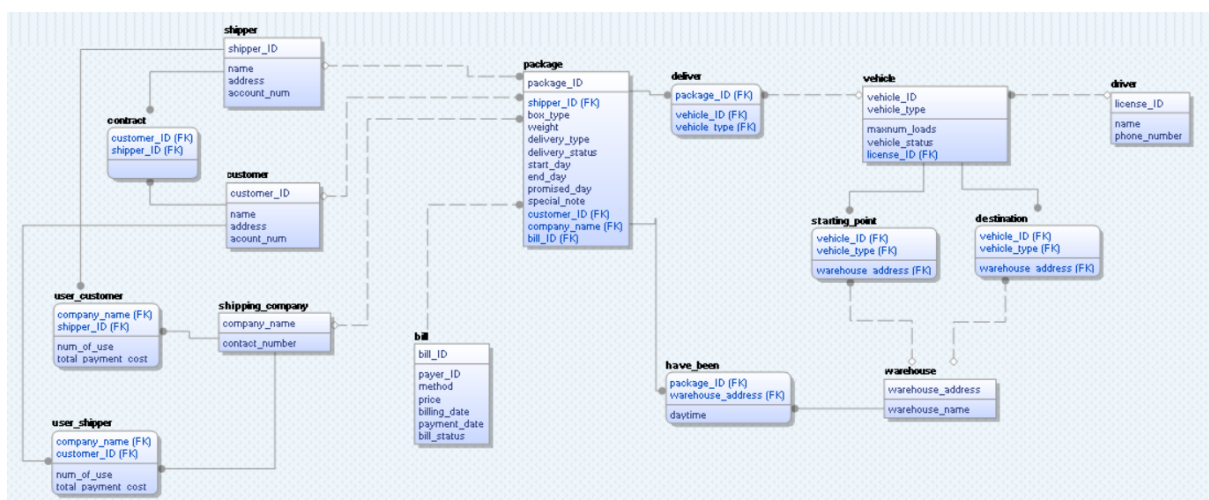
15. have been (package\_ID, warehouse\_address, daytime)

package\_ID, warehouse\_address -> daytime

총 15개의 relation의 functional dependency를 확인한 결과, 모두 좌변이 primary key인 dependency밖에 없다. primary key는 항상 superkey가 되므로, 모든 relation이 이미 BCNF를 만족한다. 따라서 BCNF로 decompose할 relation이 없다.

다만, 맨 처음에 언급했듯이, bill과 package의 relationship 변경의 필요성이 보여서 이를 수정하였고, 이에 따라 다시 E-R model을 디자인하였다.

### 3. Schema Diagram



## 4. Query

<make\_table.txt>

ERwin으로 만든 schema diagram을 참고하여 각 entity에 대한 table을 만들었다. create를 이용하였고, 각 attribute마다 어떤 type을 가져야 하는지 지정해 주었다.

<insert\_value.txt>

다양한 data에 대해 query 문을 확인하고자 insert into를 이용해서 각 table마다 5개에서 20개 정도의 tuple들을 만들어서 삽입하였다. primary key가 겹치지 않도록 주의하여 생성하였다.

<20181294.cpp>

이미 제공된 코드는 따로 설명하지 않고, 새로 작성한 코드에 대해 설명하겠다.

# read input file

```
ifstream pFile("make_table.txt");
ifstream qFile("insert_value.txt");

if (pFile.is_open())
{
    string line;
    while (getline(pFile, line))
    {
        query = line.c_str();
        mysql_query(connection, query);
    }
    pFile.close();
}
else
{
    printf("file open error");
    return 1;
}

if (qFile.is_open())
{
    string line;
    while (getline(qFile, line))
    {
        query = line.c_str();
        mysql_query(connection, query);
    }
}
```

```

    }
    qFile.close();
}
else
{
    printf("file open error");
    return 1;
}

```

우선, table을 생성하는 "make\_table.txt" 파일과 tuple을 삽입하는 "insert\_value.txt" 파일을 각각 ifstream을 이용하여 open하였다. 파일이 오픈되지 않았을 때의 에러처리는 조건문을 이용하여 진행하였다. 파일을 읽은 후, 파일 내에 있는 문자열을 한 줄씩 읽는다. 이 때, string으로 읽고 query type인 char \*로 바꿔주기 위해 c\_str() 함수를 사용하였다. 한 줄씩 읽은 문자열은 바로 mysql\_query 함수를 통해 sql에서 동작하도록 한다. 이를 위해 텍스트 파일을 작성할 때, 무조건 한 문장 안에 질의가 끝나도록 하였다. 결과적으로, 이제 table이 생성되었고, 각 table에 다양한 tuple들이 들어있다.

# menu()

```

char menu() {
    printf("----- SELECT QUERY TYPES -----WnWn");
    printf("Wt1. TYPE 1Wn");
    printf("Wt2. TYPE 2Wn");
    printf("Wt3. TYPE 3Wn");
    printf("Wt4. TYPE 4Wn");
    printf("Wt5. TYPE 5Wn");
    printf("Wt0. QUITWn");
    return _getch();
}

```

몇 번 쿼리에 대한 답을 볼지 선택하는 메뉴 함수를 따로 구현하였다. 주어진 메뉴 화면을 출력하도록 하고, \_getch() 함수를 통해 키보드로 누른 문자를 반환한다.

# case 구분

```

while (1) {
    char type_num = menu();
    char type1_num;

    system("cls");
    switch (type_num) {
        case '1':
            type1_num = _getch();

```

```

        system("cls");
        switch (type1_num) {
        case '1':
        case '2':
        case '3':
        case '0':
            break;
        }
    case '2':
    case '3':
    case '4':
    case '5':
    case '0':
        return 0;
    default:
        break;
    }
}

```

처음 menu 화면을 출력한 후 입력받는 값(menu()의 return값)을 type\_num 변수에 저장한다. 이 변수에 대해 switch 문을 작성하여 몇 번째 문제를 선택했는지 확인한다. case는 1, 2, 3, 4, 5가 있고, 0을 누르면 프로그램을 종료한다. 특별하게 1번 문제에 대해서는 3개의 꼬리 문제가 더 있기 때문에 case 1에 들어가면 \_getch함수를 실행시켜서 몇 번째 꼬리 문제를 볼지 정한다. \_getch() 함수로 입력받은 값을 type1\_num 변수에 저장하고, 마찬가지로 다시 switch문으로 문제번호를 구분한다.

이제 각 case에 대한 코드를 자세하게 보겠다.

# case 1 - 1 (type 1-1)

문제 : All customers who had a package on the truck at the time of the crash(1721번 트럭에 택배를 보낸 모든 customer를 출력한다. )

코드 설명에 앞서, 필자 table에서의 shipper가 문제에서의 customer이고, table에서의 customer는 문제에서 recipient임을 밝힌다. 번역 과정에서 혼란이 있었다.

query에 들어가는 부분 말고는 화면을 구현하기 위해 필요한 코드이므로 생략하고, query문에 대해 보겠다.

```

query = "select package.shipper_ID, shipper.name
        from package, deliver, shipper
        where package.shipper_ID = shipper.shipper_ID
          and deliver.vehicle_ID = '1721'

```

```
and deliver.vehicle_type = 'truck'

and deliver.package_ID = package.package_ID;";
```

우선 화면 상에 shipper(customer)의 ID와 shipper(customer)의 name을 출력하도록 하였다. 택배를 보낸 고객을 확인하라는 문제이기 때문에 신원정보인 ID와 name을 select하였다. 고려한 table은 총 3개이다. 우선, 1721 truck이 배달하고 있는 package를 확인하기 위해 deliver table을 선택했다. 다음은, 그 택배를 보낸 고객을 선택해야 하기 때문에 package table과 shipper table을 함께 사용해서 고객의 정보를 확인하고자 하였다. 결국, deliver.vehicle\_ID와 deliver.vehicle\_type을 이용해서 1721 truck임을 체크하였고, deliver.package\_ID와 package.package\_ID를 이용해서 트럭이 운송하고 있던 택배 ID를 확인하고, package.shipper\_ID = shipper.shipper\_ID를 이용해서 그 택배를 보낸 shipper의 ID를 확인하였다.

결과적으로 1721 truck이 운송중인 택배를 보낸 shipper의 정보를 얻을 수 있다.

# case 1 - 2 (type 1-2)

문제 : All recipients who had a package on the truck at the time of the crash(1721번 트럭에 택배를 보낸 모든 recipient를 출력한다. )

```
query = "select package.customer_ID, customer.name
        from package, deliver, customer
        where package.customer_ID = customer.customer_ID
              and deliver.vehicle_ID = '1721'
              and deliver.vehicle_type = 'truck'
              and deliver.package_ID = package.package_ID; "
```

shipper를 출력했던 1-2문제와 거의 유사하다. 실제로 스키마 상 shipper와 customer가 같은 table 형태를 갖고 있기 때문에, ID의 이름만 shipper\_ID에서 customer\_ID로 바꿔주고, from 에서 참고하는 table을 shipper에서 customer로 바꿔주면 결과를 얻을 수 있다.

결과적으로 1721 truck이 운송중인 택배를 받기로 예정된 customer의 정보를 얻을 수 있다.

# case 1 - 3 (type 1-3)

문제 : The last successful delivery by that truck prior to the crash



우선 1-3번 문제에 대해서는 해석되는 의미가 많아서 어떻게 이해하고 쿼리를 작성해야 할지 선택해야 했다. last successful delivery의 의미를 1721 truck이 이전에 택배를 고객에게 성공적으로 배송완료한 것으로 봐야 할 지, 아니면 현재 경로의 출발지인 창고에 도착했던 것을 성공적인 배송으로 봐야 할 지 중에 후자를 선택하였다. 즉, 1721 truck이 마지막으로 안전하게 도착했던 창고의 정보를 출력하였다.

```
query = "select warehouse.warehouse_address, warehouse.warehouse_name
        from start_point, warehouse
        where start_point.vehicle_ID = '1721'
              and start_point.vehicle_type = 'truck'
              and start_point.warehouse_address = warehouse.warehouse_address; ";
```

결과적으로 출력해야 할 것은 1721 truck의 출발지 창고의 정보가 된다. 그래서 각 vehicle의 출발지를 저장하는 table인 start\_point와 창고 정보를 저장하는 table warehouse를 이용하였다. 우선 start\_point table에서 1721 truck의 정보를 얻기 위해 start\_point.vehicle\_ID = '1721' and start\_point.vehicle\_type = 'truck'이라고 작성하였고, 이 출발지 warehouse의 primary key인 warehouse\_address를 서로 비교해서 그 특정 창고의 정보를 warehouse table에서 출력하였다. 창고의 정보는 창고의 주소와, 창고 이름이 있다.

결과적으로, 1721 트럭이 마지막으로 배송했던(가장 최근에 출발해온) warehouse의 정보를 얻을 수 있다.

# case2 (type 2)

문제 : the customer who has shipped the most packages in th past year

```
printf("Which Year? : ");
getline(cin, type2_year);

type2_sub = "with cnt_list(ID, value) as
            (select shipper_ID, count(*) as cnt
             from package
             where year(package.start_day) = ' ' + type2_year +
             '' group by shipper_ID),
            max_cnt(value) as(
            select max(value)
```

```

        from cnt_list)

select distinct shipper_ID, max_cnt.value as count
from package, max_cnt, cnt_list
where cnt_list.value = max_cnt.value and ID = shipper_ID; ";
query = type2_sub.c_str();

```

이 문제에서는 특정 연도를 사용자 입력으로 받아야 한다. getline 함수를 이용해서 입력으로 받은 연도를 string type type2\_year에 저장하였다. 이를 바로 문자열에 적용하기 위해 우선 string type type2\_sub에 해당 연도와 쿼리문을 합쳐서 저장하였고, 나중에 c\_str() 함수를 이용해서 이를 query에 저장하였다.

이제 쿼리문을 보면, with로 두 개의 temporary relation을 만들었다. cnt\_list(ID, value)는 package table에서의 정보를 선택한다. 입력으로 들어간 연도에 출발하는(start\_day) 택배를 먼저 확인하고, 이 개수를 count를 이용해서 센다. 결국 group by를 이용해서 각 shipper마다 해당 연도에 보낸 package의 개수를 저장한다. max\_cnt(value)는 방금 만든 cnt\_list에서 package 개수의 최댓값을 저장한다. 마지막으로 원본 쿼리에서는 cnt\_list.value = max\_cnt.value로 최댓값을 갖는 (ID, value) tuple을 찾는다.

결과적으로 해당 연도에 보낸(start day) 택배가 가장 많은 shipper와 그 개수를 얻을 수 있다.

### # case3 (type 3)

문제 : the customer who has spent the most money on shipping in the past year.

여기도 case2에서와 마찬가지로 사용자 입력으로 연도를 하나 받고, 별도의 string 변수 type3\_year에 이를 저장하였다. 위와 동일하게 string 타입 type3\_sub에 연도를 적용한 쿼리문을 모두 저장하고, c\_str() 함수를 이용해서 query에 저장한다. case3와 중복되는 내용이기 때문에 코드는 생략하겠다.

```

type3_sub = "with 2022_sum_price(ID, sum) as
            (select shipper_ID, sum(bill.price) as sum
            from package, bill
            where package.bill_ID = bill.bill_ID
            and year(package.start_day) = '" + type3_year +
            "' group by shipper_ID ),
            2022_top_price(value) as

```

```

        (select max(sum)
        from 2022_sum_price )

select ID, sum
from 2022_sum_price, 2022_top_price
where sum = value; ";

```

전반적인 코드 내용도 case 2와 비슷하다. 우선 with를 사용해서 temporary relation 2개를 생성한다. 2022\_sum\_price(ID, sum) (초기에 문제를 착각해서 이름을 2022로 붙였다. 내용에는 이상 없다)는 package와 bill 2개의 table에서 각각 package의 start day 연도가 해당 연도인 package를 선택하고, 그에 대응되는 bill 정보를 가져온다. 이를 group by shipper로 묶고, sum(bill.price)를 통해 각 shipper의 해당 연도 결제 금액 총액을 저장한다. 2022\_top\_price(value)에는 방금 만든 2022\_sum\_price의 max(sum)을 저장한다. 즉, 금액 중에 최고 금액을 하나 저장한다. 마지막으로 위에서 만든 두 table을 이용해 최고 금액을 value로 갖는 ID를 찾고, 이를 sum과 함께 저장한다.

결과적으로, 해당 연도에 택배 배송으로 결제한 금액이 가장 높은 shipper와 그 금액을 얻을 수 있다.

#### # case4 (type 4)

문제 : the packages that were not delivered within the promised time

```

query = "select package_ID, company_name, start_day, end_day, promised_day
        from package
        where package.end_day > package.promised_day; ";

```

package table을 만들 때, start day, end day, promised day를 각각 만들어 두었던 것이 도움이 되었다. 셋 다 type은 datetime을 이용하였고, start day는 택배가 출발한 날짜와 시간, end day는 택배가 도착한 날짜와 시간, promised day는 예정 배송 날짜와 시간을 저장한다. 만약 택배 배송이 아직 완료되지 않았다면 end\_day에 null 값이 저장된다. NOT delivered within the promised time은 택배가 예정 시간보다 늦게 배송된 것을 의미한다. 각 속성의 type이 datetime이기 때문에 대소 비교 operation으로 시간의 전후를 확인할 수 있다. end day > promised day는 end day가 더 늦다는 의미이므로, 문제에서 요구하는 tuple을 찾기 적합하다. data가 출력되었을 때 직접 확인할 수 있도록, 출력 시 택배 ID, 회사 이름, 출발시각, 도착시각, 예정시각을 모두 출력하도록 하였다.

결과적으로, 예정 시간보다 늦게 배송된 택배들의 정보를 얻을 수 있다.

# case5 (type 5)

문제 : Generate the bill for each customer for the past month.

이 문제에서는 특정 달을 물어보았기 때문에 사용자 입력으로 받아야 하는 게 하나 더 늘어난다. 해당 연도와 달을 모두 받아야 한다. 역시 이는 마찬가지로 string type 변수 type5\_year와 type5\_month에 저장하였고, string type typ5\_sub에 이 둘을 적용한 쿼리문을 작성하고, c\_str() 함수를 이용해서 query에 저장하였다.

```
type5_sub = "select package.shipper_ID, shipper.address, bill.method, bill.billing_date,
bill.price
            from package, bill, shipper
            where month(bill.billing_date) ='" + type5_month +
                "' and year(bill.billing_date) = '" + type5_year +
                "'and package.bill_ID = bill.bill_ID
                and package.shipper_ID = shipper.shipper_ID";
```

쿼리는 간단하다. 우선 해당 연도와 달에 결제된 bill table에서의 tuple을 먼저 찾는다. 그 tuple과 대응되는 package table에서의 tuple도 같이 잡는다. 이 과정에서 package.bill\_ID = bill.bill\_ID를 이용하였다. 이번엔 그 tuple에 있는 shipper의 정보를 얻기 위해 package.shipper\_ID = shipper.shipper\_ID를 이용한다. 결국, 해당 연도와 달에 결제한 택배를 보낸 shipper와 택배, bill 정보까지 모두 얻었다. 이제 무엇을 출력하느냐만 선택하면 되는데, 필자는 shipper의 ID, shipper의 주소, 결제 방법, 결제 날짜, 결제 금액을 출력하였다. 여기서, 또 고민이었던 점은, 한 사람이 결제한 모든 금액을 합쳐서 한 사람은 하나의 tuple로만 나타내어야 하는 것이었는데, 다양한 결제 방법을 보여주기 위해, 여러 번 결제한 사람은 shipper\_ID와 shipper\_address는 중복해서 출력되도록 하였다.

결과적으로, 해당 연도와 달에 택배를 결제한 shipper의 정보와 결제 정보를 얻을 수 있다.

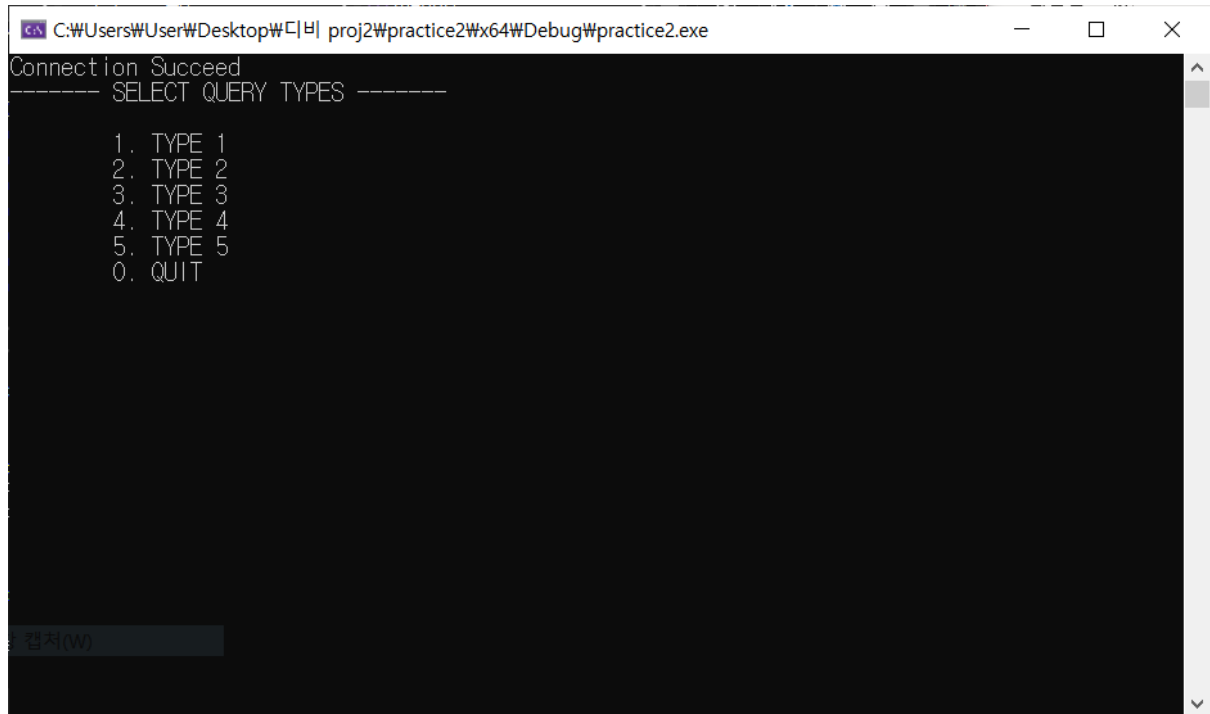
# extra

보다 깔끔하게 콘솔 창을 나타내기 위해 system("cls")를 이용해서 이전 화면이 없어지도록 하였다.

마찬가지로 보다 깔끔한 출력 화면을 나타내기 위해 변수별로 출력 시 차지하는 문자열 길이를 정해주었다. (printf("%17s %14s\n", sql\_row[0], sql\_row[1]));

## 5. 결과 출력화면

### 1. 초기화면 (메뉴)



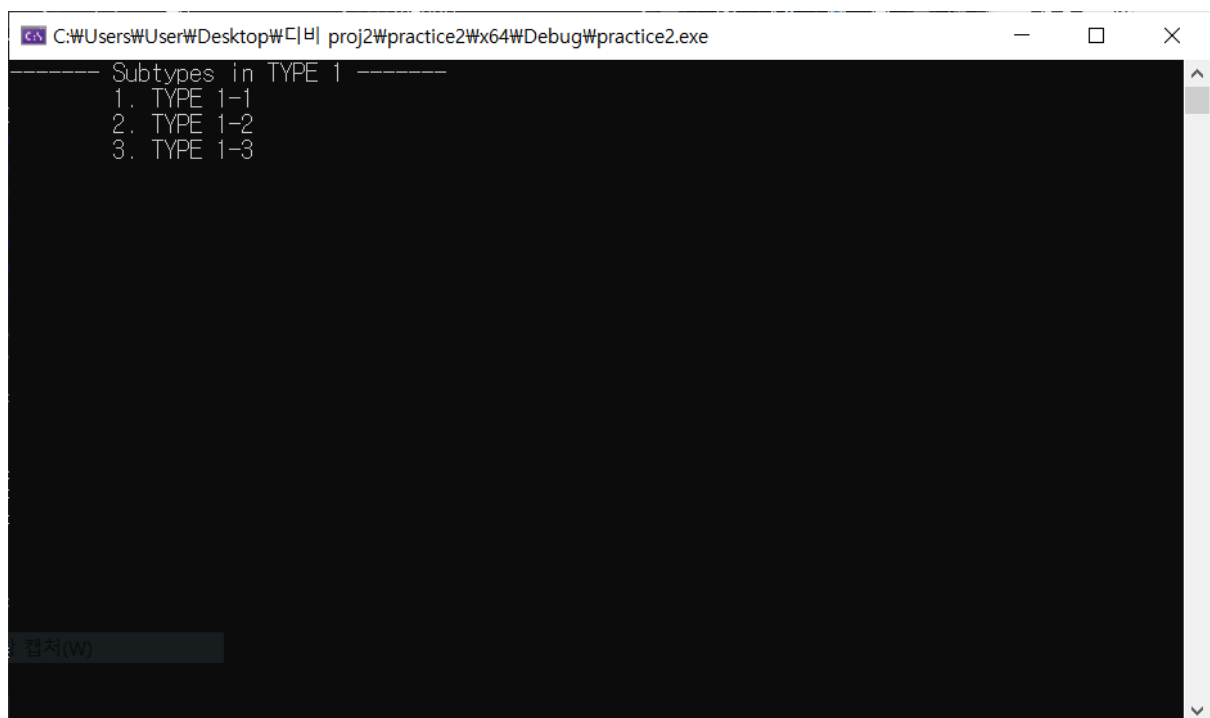
The screenshot shows a Windows command prompt window titled "C:\Users\User\Desktop\디비 proj2\practice2\Debug\practice2.exe". The output text is as follows:

```
Connection Succeed
----- SELECT QUERY TYPES -----

1. TYPE 1
2. TYPE 2
3. TYPE 3
4. TYPE 4
5. TYPE 5
0. QUIT
```

At the bottom left, there is a small grey box containing the text "캡처(w)".

### 2. 1번 문제의 꼬리문제 선택 메뉴



The screenshot shows the same Windows command prompt window. The output text is as follows:

```
----- Subtypes in TYPE 1 -----
1. TYPE 1-1
2. TYPE 1-2
3. TYPE 1-3
```

At the bottom left, there is a small grey box containing the text "캡처(w)".

### 3. 1-1번 문제

```
C:\Users\User\Desktop\디비\proj2\practice2\Debug\practice2.exe
---- TYPE 1-1 ----
truck 1721 is destroyed in a crash
All customers who had a package on the truck at the time of the crash

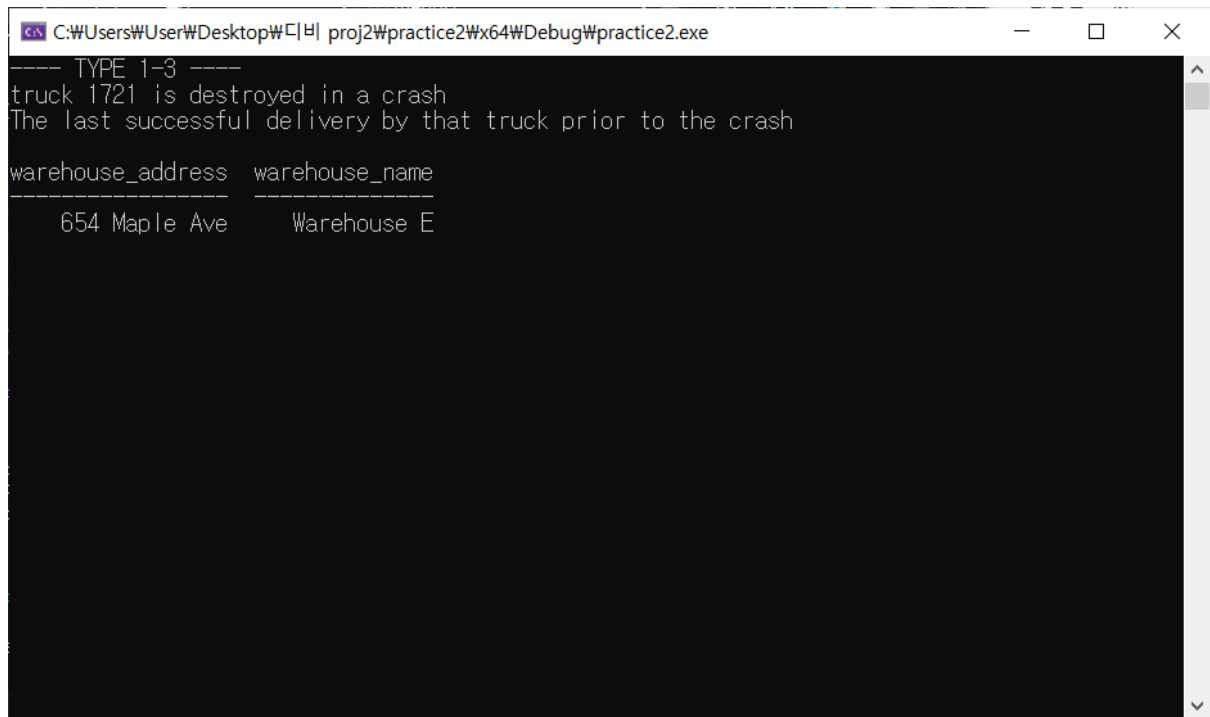
customer_ID  customer_name
-----
          S008      Sophia Hernandez
          S010      Emma Ngushipparyen
          S002              Jane Smith
```

### 4. 1-2번 문제

```
C:\Users\User\Desktop\디비\proj2\practice2\Debug\practice2.exe
---- TYPE 1-2 ----
truck 1721 is destroyed in a crash
All recipients who had a package on the truck at the time of the crash

recipient_ID  recipient_name
-----
          C012      Ava Martinez
          C014      Mia White
          C011      Daniel Kim
```

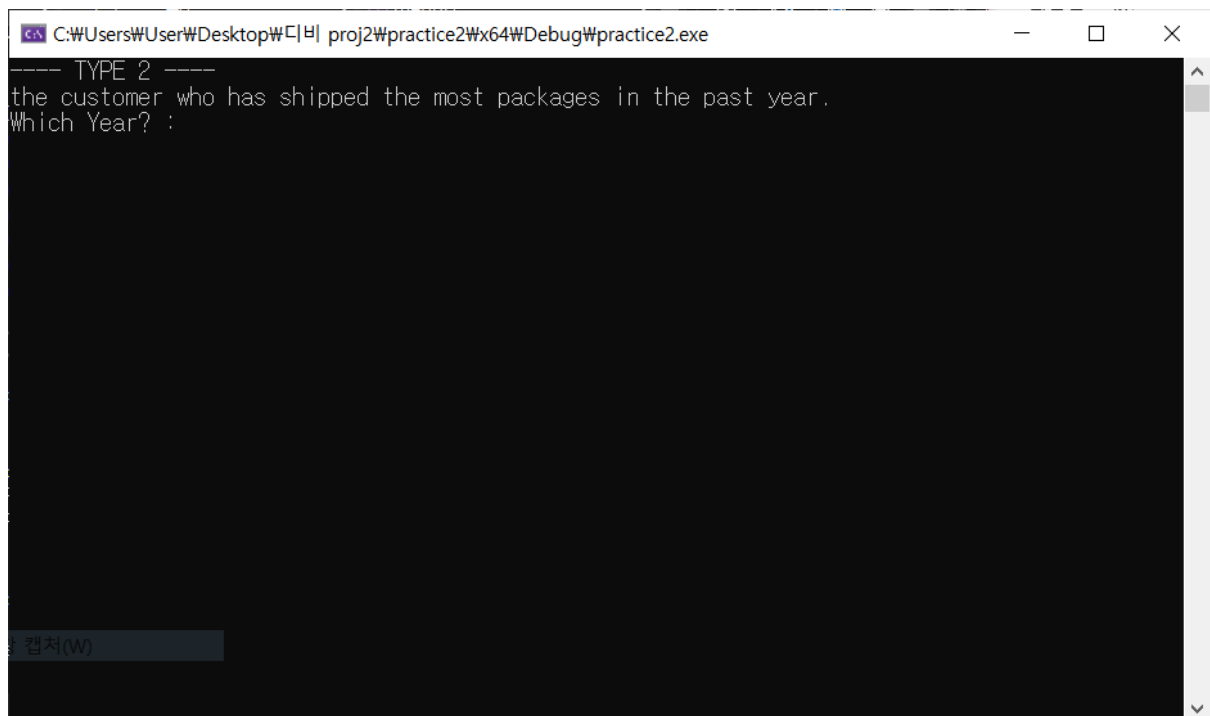
### 5. 1-3번 문제



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\User\Desktop\디비 proj2\practice2\Debug\practice2.exe. The window contains the following text:

```
---- TYPE 1-3 ----  
truck 1721 is destroyed in a crash  
The last successful delivery by that truck prior to the crash  
  
warehouse_address  warehouse_name  
-----  
654 Maple Ave      Warehouse E
```

### 6. 2번 문제 (입력창)

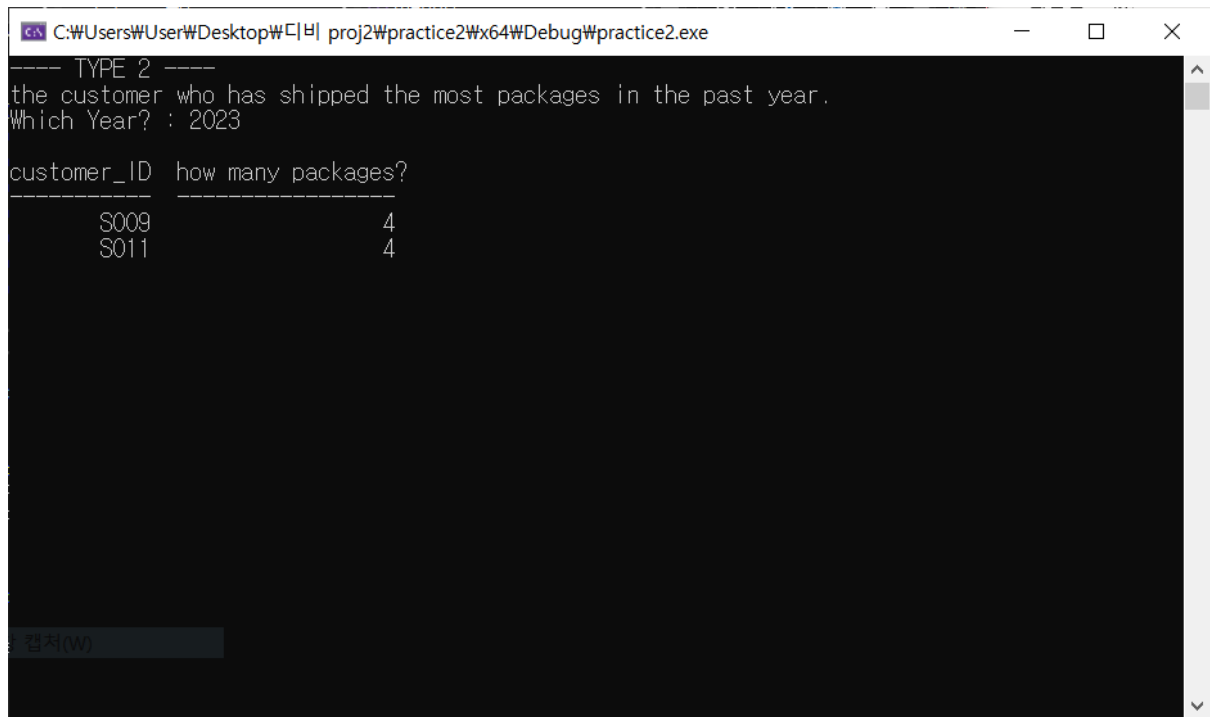


A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\User\Desktop\디비 proj2\practice2\Debug\practice2.exe. The window contains the following text:

```
---- TYPE 2 ----  
the customer who has shipped the most packages in the past year.  
Which Year? :
```

At the bottom of the window, there is a dark grey rectangular box containing the text "캡처(w)".

## 7. 2번 문제 (결과 1)

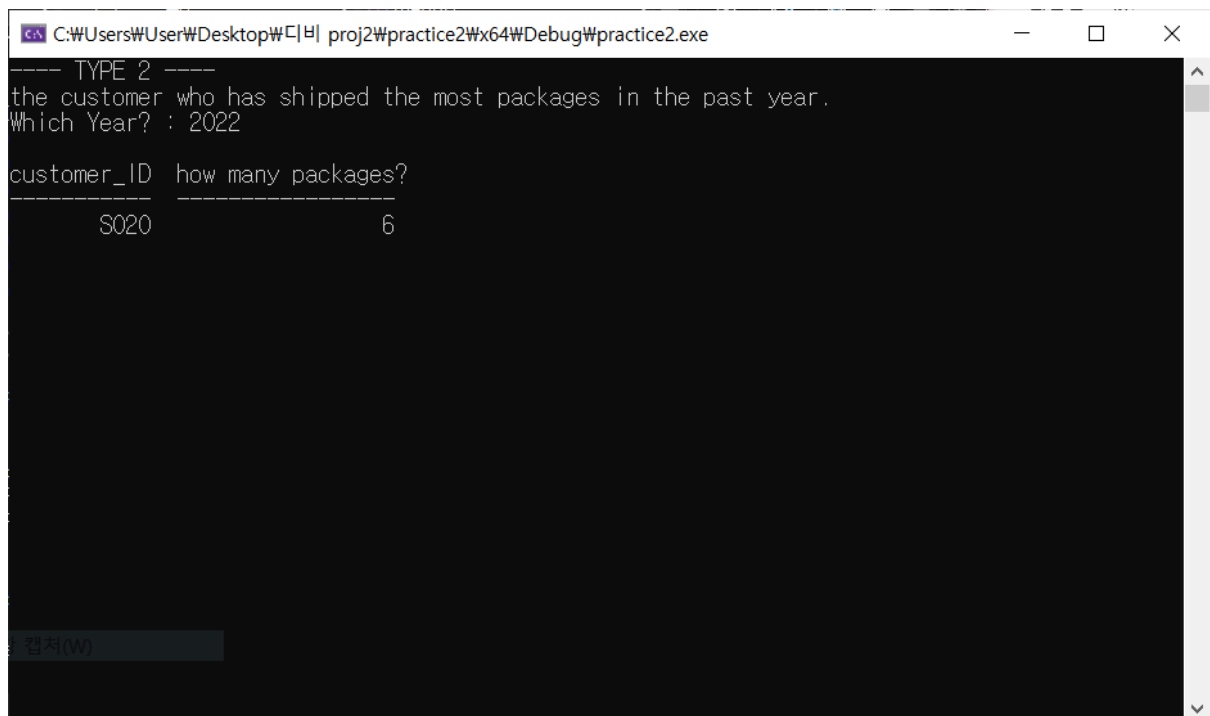


```
C:\Users\User\Desktop\디비 proj2\practice2\Debug\practice2.exe
---- TYPE 2 ----
the customer who has shipped the most packages in the past year.
Which Year? : 2023

customer_ID  how many packages?
-----
          S009                4
          S011                4
```

캡처(w)

## 8. 2번 문제 (결과2)



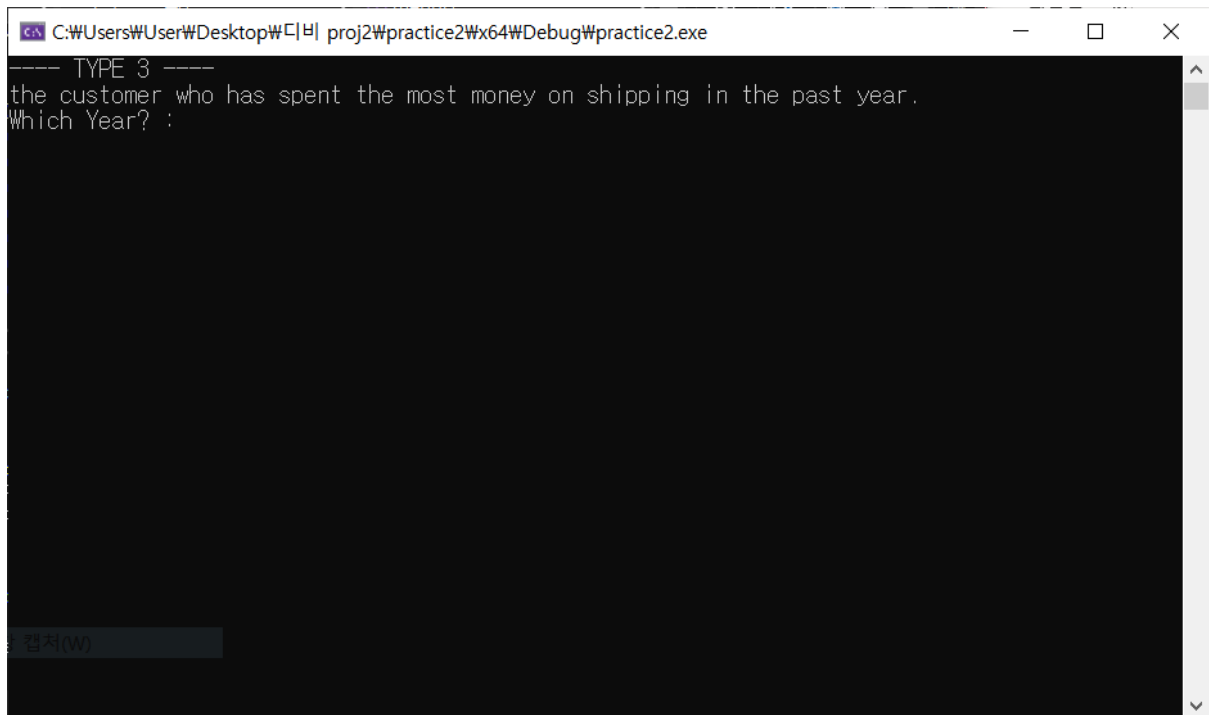
```
C:\Users\User\Desktop\디비 proj2\practice2\Debug\practice2.exe
---- TYPE 2 ----
the customer who has shipped the most packages in the past year.
Which Year? : 2022

customer_ID  how many packages?
-----
          S020                6
```

캡처(w)

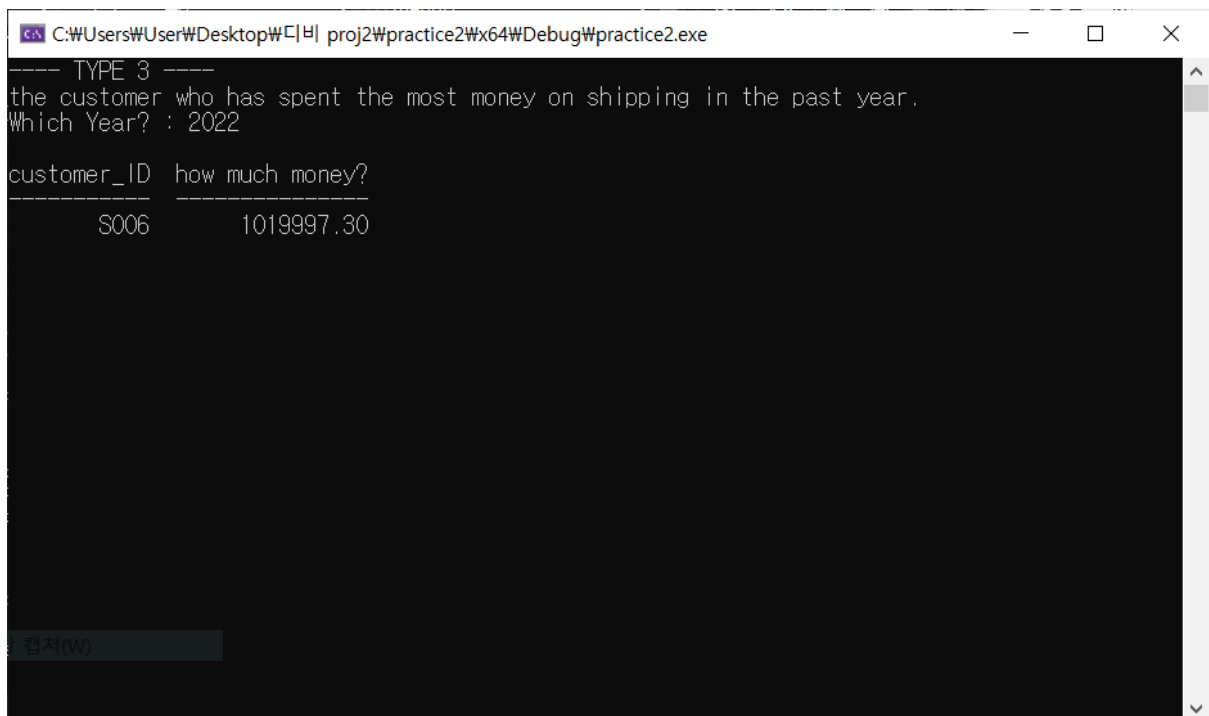


### 9. 3번 문제 (입력창)



```
C:\Users\User\Desktop\디비 proj2\practice2\x64\Debug\practice2.exe
---- TYPE 3 ----
the customer who has spent the most money on shipping in the past year.
Which Year? :
캡처(w)
```

### 10. 3번 문제 (결과 1)



```
C:\Users\User\Desktop\디비 proj2\practice2\x64\Debug\practice2.exe
---- TYPE 3 ----
the customer who has spent the most money on shipping in the past year.
Which Year? : 2022
customer_ID  how much money?
-----
      S006      1019997.30
캡처(w)
```

### 11. 3번 문제 (결과 2)

```
C:\Users\User\Desktop\디비 proj2\practice2\64\Debug\practice2.exe
---- TYPE 3 ----
the customer who has spent the most money on shipping in the past year.
Which Year? : 2023

customer_ID  how much money?
-----
          S020          9999.10
```

### 12. 4번 문제

```
C:\Users\User\Desktop\디비 proj2\practice2\64\Debug\practice2.exe
---- TYPE 4 ----
the packages that were not delivered within the promised time.

package_ID  company_name  start_day  delivered_day  promised_day
-----
          P006          Logen  2023-06-01 10:30:00  2023-06-03 14:45:00  2023-06-02 14:45:00
          P009           GS  2023-06-01 10:00:00  2023-06-03 14:15:00  2023-06-02 14:15:00
          P014           CJ  2023-06-01 08:45:00  2023-06-03 11:00:00  2023-06-02 11:00:00
          P018          Logen  2022-06-01 10:30:00  2022-06-03 14:45:00  2022-06-02 14:45:00
          P019           CJ  2022-06-01 08:00:00  2022-06-02 08:00:00  2022-06-01 12:15:00
          P020          Wooche  2022-06-01 07:15:00  2022-06-03 09:30:00  2022-06-02 09:30:00
          P333          Wooche  2021-06-01 07:15:00  2021-06-03 09:30:00  2021-06-01 09:30:00
          P444          Wooche  2022-06-01 07:15:00  2022-06-03 09:30:00  2022-06-01 09:30:00
```

### 13. 5번 문제 (입력창)

```
C:\Users\User\Desktop\디비 proj2\practice2\64\Debug\practice2.exe
---- TYPE 5 ----
Generate the bill for each customer for the past month. Consider creating several types of
bills.
Which Year and Month? (ex. 2023 06 ) :
```

### 14. 5번 문제 (결과 1)

```
C:\Users\User\Desktop\디비 proj2\practice2\64\Debug\practice2.exe
---- TYPE 5 ----
Generate the bill for each customer for the past month. Consider creating several types of
bills.
Which Year and Month? (ex. 2023 06 ) : 2023 5

customer_ID  customer_address  bill_method  billing_date  bil_price
-----
S007         234 Birch St      Cash         2023-05-20 14:15:00  100.10
S007         234 Birch St      Credit Card  2023-05-20 14:15:00  300.10
S008         567 Walnut St     Cash         2023-05-10 14:15:00  1012.10
S009         890 Spruce St     Cash         2023-05-11 14:15:00  11.10
S009         890 Spruce St     Cash         2023-05-20 14:15:00  22.10
S009         890 Spruce St     Credit Card  2023-05-09 14:15:00  310.10
S010         432 Fir St        Credit Card  2023-05-30 14:15:00  900.10
S011         765 Ash St        Cash         2023-05-20 14:15:00  810.10
S011         765 Ash St        Credit Card  2023-05-20 14:15:00  420.10
S011         765 Ash St        Bank transfer 2023-05-08 14:15:00  440.10
S011         765 Ash St        Cash         2023-05-01 14:15:00  400.10
S012         098 Oak St        Cash         2023-05-20 14:15:00  500.10
S013         321 Pine St       Cash         2023-05-20 14:15:00  20.10
```

## 15. 5번 문제 (결과 2)

```
C:\Users\User\Desktop\디비 proj2\practice2\wx64\Debug\practice2.exe
---- TYPE 5 ----
Generate the bill for each customer for the past month. Consider creating several types of
bills.
Which Year and Month? (ex. 2023 06 ) : 2023 04

customer_ID  customer_address  bill_method  billing_date  bil_price
-----
S001         123 Main St      Credit Card  2023-04-01 12:00:00  250.75
S003         789 Oak St       Cash        2023-04-02 10:30:00  150.50
S008         567 Walnut St    Credit Card  2023-04-03 11:45:00  198.25
S010         432 Fir St       Cash        2023-04-04 14:20:00  180.00
S002         456 Elm St       Credit Card  2023-04-05 16:30:00  275.80
S004         321 Pine St      Cash        2023-04-06 09:15:00  210.00
S009         890 Spruce St    Credit Card  2023-04-07 13:40:00  350.20
S006         987 Maple St     Cash        2023-04-08 15:20:00  320.50
S019         432 Spruce St    Credit Card  2023-04-09 11:10:00  175.30
S018         890 Walnut St    Cash        2023-04-10 14:45:00  198.75
S006         987 Maple St     Credit Card  2023-04-11 16:55:00  285.20
S010         432 Fir St       Cash        2023-04-12 10:25:00  220.00
S004         321 Pine St      Credit Card  2023-04-13 12:50:00  390.60
S007         234 Birch St     Cash        2023-04-14 14:30:00  280.75
S006         987 Maple St     Cash        2023-04-20 14:15:00  9999.10
```

## 6. Conclusion

이렇게 실제 쿼리문까지 작성하면서 택배 시스템을 구현하는 프로젝트를 마무리하였다. 이전에 해왔던 타 과목 과제들과는 달리, 스스로 무언가를 만들어내야 하는 시스템이라서 초반에는 막막하고 진전이 없었지만, diagram을 완성하고, mysql에 직접 코드를 작동시켜보며 수업시간에 배웠던 내용이 머릿속에 더 깊게 저장되었다. 특히 문제에 대한 쿼리문을 작성하는 과정에서 중간고사 기간에 공부했던 내용을 다시 한 번 되짚어보게 되어 오래 기억하는 데 도움이 되었다.

다만 아쉬운 점은, BCNF로 decompose하는 알고리즘을 열심히 공부하였지만 실제로 적용을 못했다는 점이다. 기존에 table을 잘 구현해둔 덕분이긴 하지만, 공부 면에서는 아쉬운 것이 사실이다.

생각보다 많은 시간을 투자해야 하는 프로젝트였지만, 원하는 결과를 얻을 수 있어서 아주 만족스러운 경험이었다. DB를 언제 다시 다루게 될지는 모르겠지만, 분명 이후 공부에 큰 도움을 줄 것이라고 생각한다.

