데이터베이스시스템

CSE4110-01

[Project2]

서강대학교 컴퓨터공학과

학번 : 20191583

이름 : 김태곤

I. BCNF Decomposition

Logical Schema Diagram은 기존 Project1과 크게 달라진 점은 없다. 다만 보다 정확한 내용을 담아낼 수 있게 Shipment에 Hazardous, International을 추가하였고, Location의 timestamp를 Start_Time과 End_Time으로 나누었다. BCNF를 판별하면 다음과 같다.

1) Customer

Functional Dependency : Customer_ID → Name, Phone_Number, Address, Contract

Primary Key : Customer_ID

Primary Key를 구성하는 속성인 Customer_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다. Primary key가 Super key이므로 모든 Functional Dependency를 만족시키며, 다른 속성들이 Primary key에 종속되어 있다.

2) Recipient

Functional Dependency : Recipient_ID → Name, Phone_Number, Address

Primary Key : Recipient_ID

Primary Key를 구성하는 속성인 Recipient_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다. Primary key가 Super key이므로 모든 Functional Dependency를 만족시키며, 다른 속성들이 Primary key에 종속되어 있다.

3) Shipment

Functional Dependency : Shipment_ID → Items, Order_Time, Receive_Time, international, Hazardous, Customer_ID, Recipient_ID

Primary Key : Shipment_ID

Primary Key를 구성하는 속성인 Shipment_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다. Primary key가 Super key이므로 모든 Functional Dependency를 만족시키며, 다른 속성들이 Primary key에 종속되어 있다.

4) Bill

Functional Dependency : Bill_ID → Pay_Method, Price, Billing_Date, State

Primary Key : Bill_ID

Primary Key를 구성하는 속성인 Bill_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다. Primary key가 Super key이므로 모든 Functional Dependency를 만족시키며, 다른 속성들이 Primary key에 종속되어 있다.

5) Ship_Billl

Functional Dependency : Shipment_ID → Bill_ID

Primary Key : Shipment_ID

Primary Key를 구성하는 속성인 Shipment_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다.

6) Bill_Cus

Functional Dependency : Bill_ID → Customer_ID

Primary Key : Bill_ID

Primary Key를 구성하는 속성인 Bill_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다.

7) Location

Functional Dependency : Shipment_ID, Position_Type, Name_ID → Start_Time, End_Time, State

Primary Key : Shipment_ID, Position_Type, Name_ID

Primary Key를 구성하는 속성인 Shipment_ID, Position_Type, Name_ID 는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다. Primary key가 Super key이므로 모든 Functional Dependency를 만족시키며, 다른 속성들이 Primary key에 종속되어 있다.

8) Service

Functional Dependency : X

Primary Key : Package_Type, Weight, TimeLiness

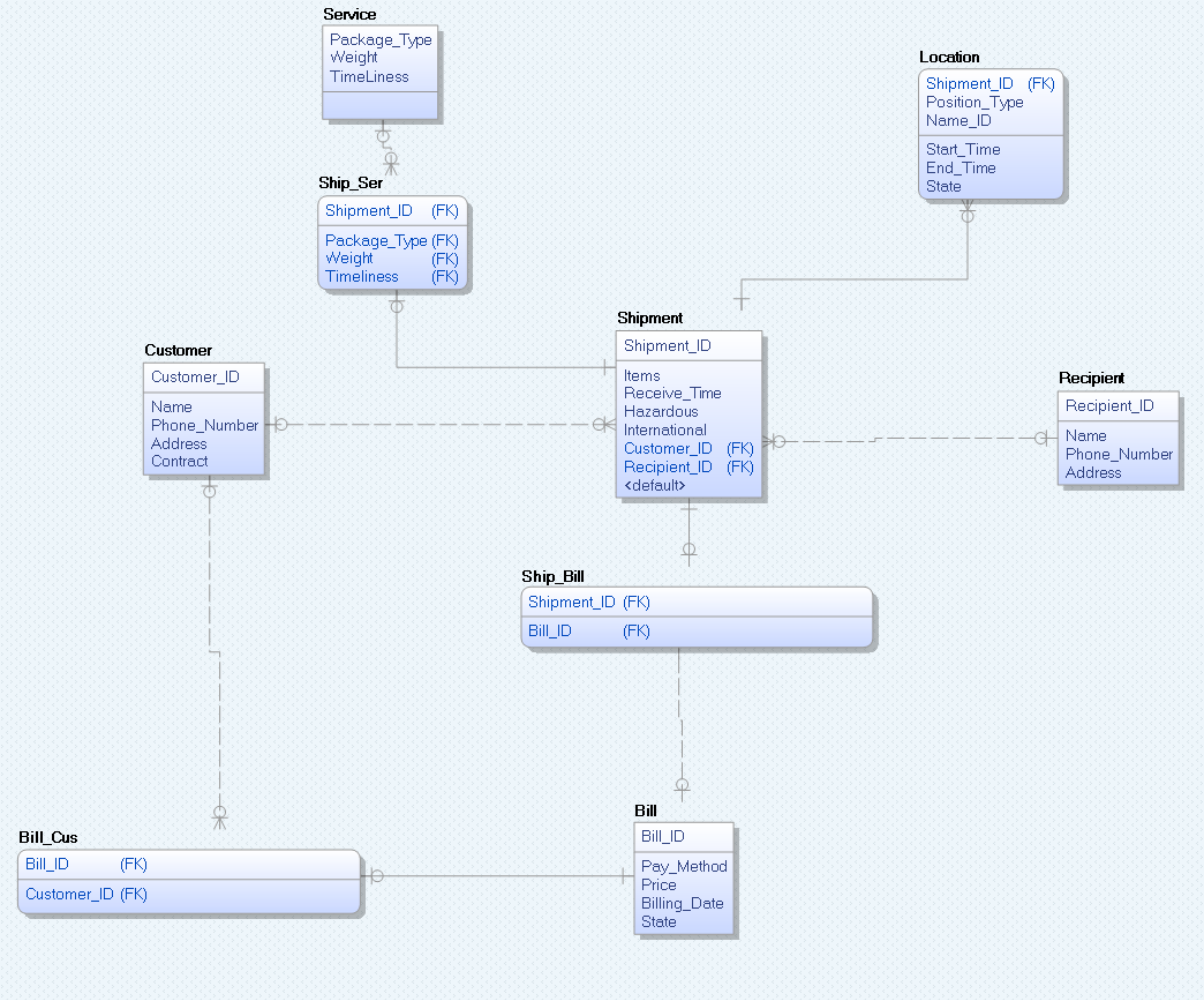
Primary Key를 구성하는 속성인 Package_Type, Weight, TimeLiness는 다른 속성들에 대해 Functional Dependency를 가지고 있지 않다.

9) Ship_Ser

Functional Dependency : Shipment_ID → Package_Type, Weight, Timeliness

Primary Key : Shipment_ID

Primary Key를 구성하는 속성인 Shipment_ID는 모든 속성에 대해 Functional Dependency를 만족하므로 BCNF를 만족한다



II. Physical Schema Diagram

전반적인 데이터의 값은 NULL값이 들어오지 못하도록 설계하였다. 이중 Shipment의 Receive_Time은 사고가 날 경우 도착한 시간이 없을 수 있으므로 NULL값을 허용하였다. 각 속성 별 데이터 타입은 다음과 같이 정의하였다.

1) Shipment

Shipment_ID : 배송에 대한 ID이므로 int형으로 설정하였다.

Items : 배송 물건에 대한 이름이므로 char(18)로 설정하였다.

Order_Time : 주문 시간에 대한 정보이므로 시간을 나타내는 datetime으로 설정하였다.

Receive_Time : 도착 시간에 대한 정보로 시간을 나타내는 datetime으로 설정하였다.

International : 국제 배송에 대한 정보로 예/아니요만 필요하므로 bit로 설정하였다.

Hazardous : 위험 주의 물건에 대한 정보로 예/아니요만 필요하므로 bit로 설정하였다.

2) Customer

Customer_ID : 고객에 대한 ID이므로 int형으로 설정하였다.

Name : 고객 이름에 대한 정보이므로 char(18)로 설정하였다.

Phone_Number : 초기에는 int형으로 생각하였나 int형에는 나타낼 수 있는 자릿수에 제한이 있어 한국 전화번호를 다 나타낼 수 없으므로 문자형인 char(18)로 설정하였다.(검색결과 실제로도 문자열로 나타내는 곳이 많았다.)

Address : 주소에 대한 정보로 실제로는 더 긴 문자열이 필요하지만, 실습에서 지역만 간단하게 나타내기 위해 char(18)로 설정하였다.

Contract : 정기 결제에 대한 구독 내용으로 예/아니요만 필요하므로 bit로 설정하였다.

3) Recipient

Recipient_ID : 고객에 대한 ID이므로 int형으로 설정하였다.

Name : 고객 이름에 대한 정보이므로 char(18)로 설정하였다.

Phone_Number : 초기에는 int형으로 생각하였나 int형에는 나타낼 수 있는 자릿수에 제한이 있어 한국 전화번호를 다 나타낼 수 없으므로 문자형인 char(18)로 설정하였다.(검색결과 실제로도 문자열로 나타내는 곳이 많았다.)

Address : 주소에 대한 정보로 실제로는 더 긴 문자열이 필요하지만, 실습에서 지역만 간단하게 나타내기 위해 char(18)로 설정하였다.

4) Bill

Bill_ID : 주문 비용 청구에 대한 ID이므로 int형으로 설정하였다.

Pay_Method : 결제 방법에 대한 내용이므로 char(18)로 설정하였다.

Billing_Date : 결제 요청 시간에 대한 내용이므로 datetime으로 설정하였다.

State : 비용 청구 완료에 대한 내용으로 예/아니요만 필요하므로 bit로 설정하였다.

5) Service

Package_Type : 패키지의 포장 방법에 대한 내용이므로 char(18)로 설정하였다.

Weight : 패키지의 무게에 대한 내용이므로 float로 설정하였다.

TimeLiness : 배송 기한에 대한 내용으로 시간을 나타내는 datetime이 아닌 over night, second day, longer 등 한글로 나타내기 위해 char(18)로 설정하였다.

6) Location

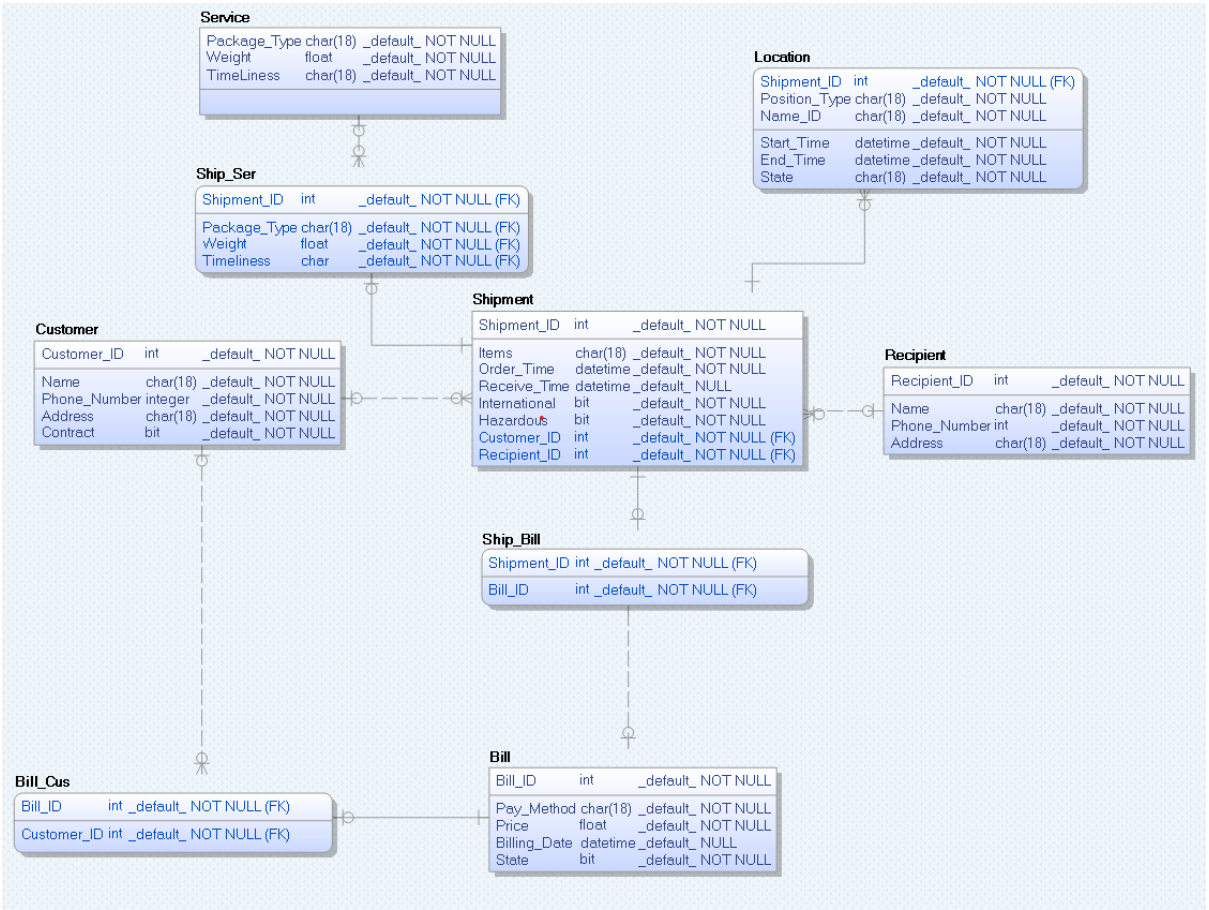
Position_Type : 현재 배송 위치에 대한 내용으로 truck, hub 등을 나타내므로 char(18)로 설정하였다.

Name_ID : 현재 배송 위치에 대한 구체적인 정보로 truck에 대한 번호, hub에 대한 위치 등 복합적으로 나타내기 위해 char(18)로 설정하였다.

Start_Time : 배송 시작 시간, 현재 위치에 머물기 시작한 시간 등을 나타내기 위해 datetime으로 설정하였다.

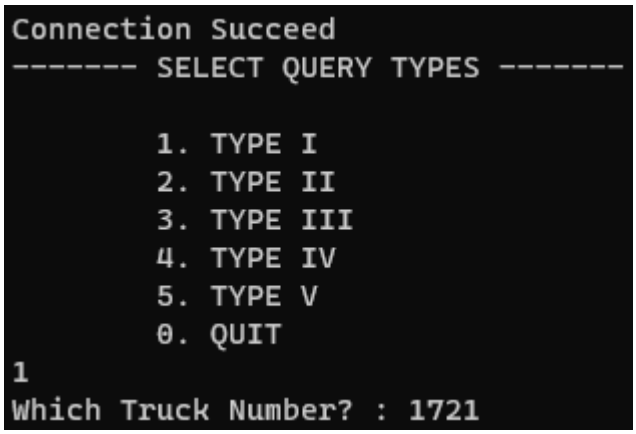
End_Time : 배송 도착 시간, 현재 위치에서 떠난 시간 등을 나타내기 위해 datetime으로 설정하였다.

State : 현재 패키지가 배송 중인지, 창고에 있는지, 도중에 사고가 발생했는지 등 상태를 나타내기 위해 char(18)로 설정하였다.



III. Queries

먼저 connection이 성공정으로 이루어지면 create_table파일과 insert_table파일을 읽어와 mysql_query를 통해 query를 넣어주었다. 이후 while문을 통해 숫자를 입력받고 각 입력한 숫자에 해당하는 query를 spirntf를 통해 입력 받은 parameter와 합쳐 query를 만들어 주었다. 이후 파일을 읽어 query를 넣어준 것과 같은 방법으로 mysql_query를 통해 입력해주었고, state를 판별해 정상적으로 query가 들어온 경우(state=0) query의 select 수에 맞게 출력해주었다.



1) Assume truck X is destroyed in a crash.

먼저 하위 질의를 처리하기 전에 truck의 number 'X'를 입력받았다.

아래 질의를 처리할 때 crash 상태의 truck X가 없다면 'truck X didn't crush'를 출력하였다.

1-1) Find all customers who had a package on the truck at the time of the crash.

```
SELECT C.Name
FROM Customer C
INNER JOIN Shipment S ON C.Customer_ID = S.Customer_ID
INNER JOIN Location L ON S.Shipment_ID = L.Shipment_ID
WHERE L.Position_Type = 'truck' AND L.Name_ID = 'X' AND L.State = 'crash'
```

충돌 당시 해당 truck이 가지고 있던 패키지를 보낸 사람의 이름을 select하였다.

```
----- Subtypes in TYPE I -----
        1. TYPE I-1.
        2. TYPE I-2.
        3. TYPE I-3.
        0. Exit.
1
---- TYPE I-1 ----

**Find all customers who had a package on the truck at the time of the crash.**
Dongheon
Taegon
Yoonhyuk
```

1-2) Find all recipients who had a package on that truck at the time of the crash.

```
SELECT R.Name

FROM Recipient R

INNER JOIN Shipment S ON R.Recipient_ID = S.Recipient_ID

INNER JOIN Location L ON S.Shipment_ID = L.Shipment_ID

WHERE L.Position_Type = 'truck' AND L.Name_ID = '%d' AND L.State = 'crash'
```

충돌 당시 해당 truck이 가지고 있던 패키지의 수신자 이름을 select하였다.

```
----- Subtypes in TYPE I -----
        1. TYPE I-1.
        2. TYPE I-2.
        3. TYPE I-3.
        0. Exit.
2
---- TYPE I-2 ----

**Find all recipients who had a package on that truck at the time of the crash.**
Mia
Ava
Sophia
```

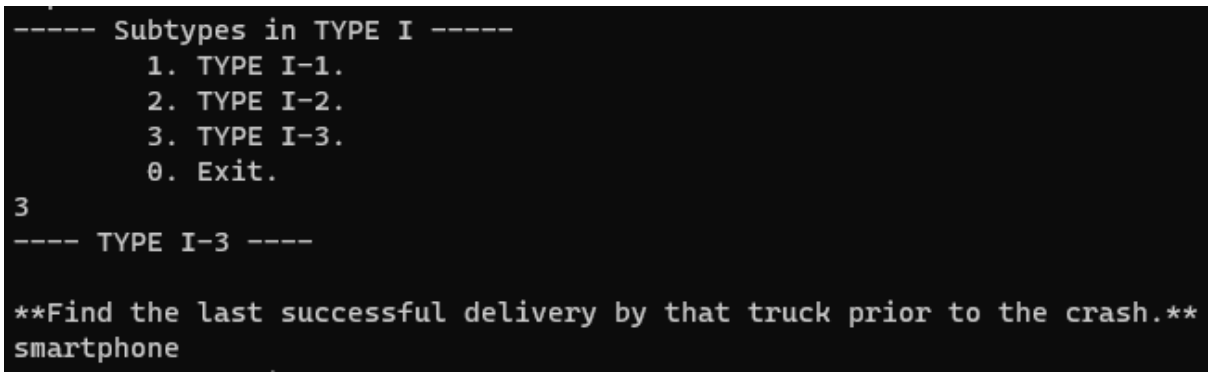
1-3) Find the last successful delivery by that truck prior to the crash.

```
SELECT S.Items

FROM Shipment S

INNER JOIN Location L ON S.Shipment_ID = L.Shipment_ID

WHERE L.Position_Type = 'truck' AND L.Name_ID = '%d' AND L.State = 'arrive'
```

```
AND L.End_Time
        = ( SELECT MAX(End_Time)
            FROM Location
            WHERE Position_Type = 'truck' AND Name_ID = '%d' AND State = 'arrive'
        AND End_Time <
            ( SELECT MAX(End_Time)
              FROM Location
              WHERE Position_Type = 'truck' AND Name_ID = '%d'
              AND State = 'crash' ) )
```

충돌 당시 시간을 찾아내고(여러 번의 사고가 발생했을 수도 있으므로 가장 최근 발생한 사고 기준으로 작성하였다.), arrive 상태에서 해당 충돌 시간보다 End_Time이 작은 것들 중 가장 큰 End_Time을 골라냈다. 이후 가장 큰 End_Time을 가진 arrive 상태의 Location 정보를 찾아내 items을 select하였다.

출력 결과는 충돌 직전에 배송된 item이 나온다.



2) Find the customer who has shipped the most packages in the past year.

Query를 처리하기 전 Year에 대한 정보를 입력 받았다. 해당 Year에서 가장 많은 배송을 요청한 고객의 이름과 배송 요청 수를 추출하였다.

```
SELECT C.Name, COUNT(S.Shipment_ID) AS Shipment_Count
FROM Customer C JOIN Shipment S ON C.Customer_ID = S.Customer_ID
WHERE YEAR(S.Order_Time) = 'Year'
GROUP BY C.Name
```

```
HAVING COUNT(S.Shipment_ID)
        = ( SELECT MAX(Shipment_Count)
          FROM( SELECT COUNT(Shipment_ID) AS Shipment_Count
                FROM Shipment
                WHERE YEAR(Order_Time) = % d GROUP BY Customer_ID ) AS T )
```

Customer 테이블과 Shipment 테이블을 Join하여, 년도에 해당하는 주문 시간을 가진 고객의 이름과 함께 출하된 물품의 수를 계산하였다. GROUP BY C.Name은 고객 이름별로 그룹화했다. HAVING COUNT(S.Shipment_ID)를 사용하여 출하된 물품의 수가 가장 많은 고객을 필터링하였다. 서브쿼리를 사용하여 Shipment 테이블에서 년도가 일치하는 주문 시간을 가진 고객별로 출하된 물품의 수를 계산하고, 그 중 가장 큰 값을 선택하였다. 이 과정을 통해 출하된 물품의 수가 가장 많은 고객의 이름과 해당 고객이 출하한 물품의 수가 출력된다.

```
------- SELECT QUERY TYPES -------

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
2
---- TYPE II ----

**Find the customer who has shipped the most packages in the past year.**
Which Year? : 2023
Juhyeon 3
```

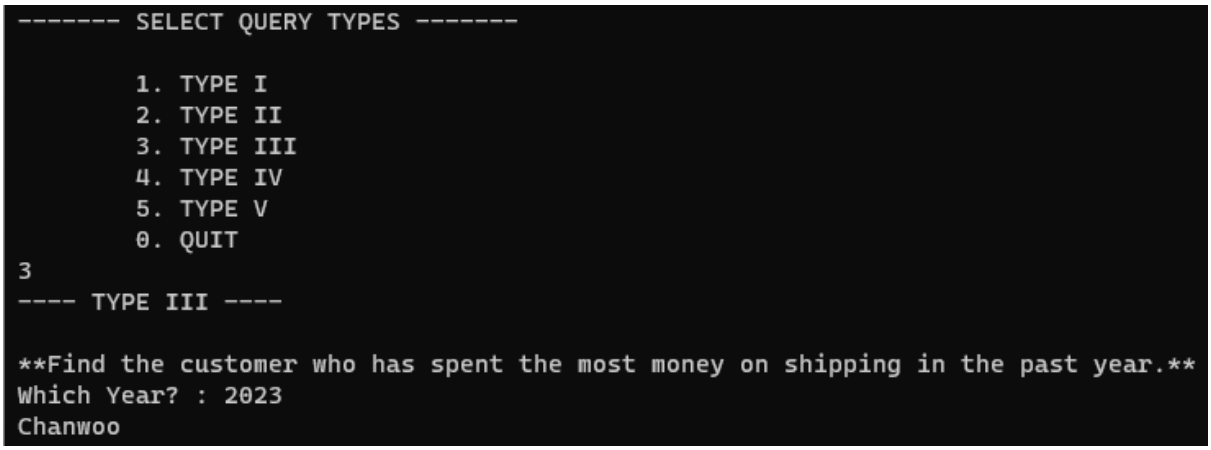
3) Find the customer who has spent the most money on shipping in the past year.

Query를 처리하기 전 Year에 대한 정보를 입력 받았다. 해당 년도에 Shipping으로 가장 많은 비용을 쓴 사람의 이름을 출력한다.

```
SELECT C.Name
FROM Customer C
JOIN Bill_Cus BC ON C.Customer_ID = BC.Customer_ID
JOIN Bill B ON BC.Bill_ID = B.Bill_ID
```

```
WHERE YEAR(B.Billing_Date) = Year

GROUP BY C.Customer_ID, C.Name

HAVING SUM(B.Price) = ( SELECT MAX(Total_Price)

        FROM( SELECT SUM(B2.Price) AS Total_Price

                FROM Bill_Cus BC2 JOIN Bill B2 ON BC2.Bill_ID = B2.Bill_ID

                WHERE YEAR(B2.Billing_Date) = Year

                GROUP BY BC2.Customer_ID ) AS T )
```

Customer, Bill_Cus, Bill 테이블을 Join하여, 년도에 해당하는 Bill의 고객 정보와 청구 금액을 가져왔다. GROUP BY C.Customer_ID, C.Name을 사용하여 고객별로 그룹화하였다. HAVING SUM(B.Price)를 사용하여 청구 금액의 합이 가장 큰 고객을 필터링하였고, 서브쿼리를 사용하여 Bill_Cus, Bill 테이블에서 해당 년도의 Bill의 고객별로 청구 금액의 합을 계산하였고, 그 중 가장 큰 값을 선택했다. 최종적으로 청구 금액의 합이 가장 큰 고객의 이름이 출력하였다.



4) Find the packages that were not delivered within the promised time.

```
SELECT S.Shipment_ID, S.Items

FROM Shipment S

JOIN Ship_Ser SS ON S.Shipment_ID = SS.Shipment_ID

WHERE TIMESTAMPDIFF(HOUR, S.Order_Time, S.Receive_Time) >

        CASE WHEN SS.Timeliness = 'overnight' THEN 24
```

WHEN SS.Timeliness = 'second day' THEN 48

WHEN SS.Timeliness = 'longer' THEN 720 END

OR S.Receive_Time IS NULL

Shipment와 Ship_Ser 테이블을 join하여 Shipment_ID와 상품 항목을 가져왔다. TIMESTAMPDIFF를 사용하여 도착 시간과 주문 시간의 차이를 계산하였고, Timeliness에 따라 시간을 지정하여 비교하였다. 또한 도착하지 못한 상품도 선택하여 최종적으로 Shipment_ID와 Items를 출력하였다.

```
------- SELECT QUERY TYPES -------

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
4
---- TYPE IV ----

**Find the packages that were not delivered within the promised time.**
1 shoes
6 bag
7 cup
10 keyboard
11 pants
12 hat
```

5) Generate the bill for each customer for the past month. Consider creating several types of bills. A simple bill: customer, address, and amount owed.

Query를 처리하기 전 Year와 Month에 대한 정보를 입력 받았다. 입력된 Year와 Month에 해당하는 Bill이 있는 경우 그에 맞는 Customer의 이름과 주소, 미지급한 금액의 합을 출력하였다.

SELECT C.Name, C.Address, SUM(CASE WHEN B.State = 0 THEN B.Price

ELSE 0 END) AS Outstanding_Amount

FROM Customer C

LEFT JOIN Bill_Cus BC ON C.Customer_ID = BC.Customer_ID

LEFT JOIN Bill B ON BC.Bill_ID = B.Bill_ID

WHERE YEAR(B.Billing_Date) = %d AND MONTH(B.Billing_Date) = %d

GROUP BY C.Customer_ID

Customer, Bill_Cus, Bill 테이블을 join하여 고객의 이름, 주소, 청구서의 가격을 가져왔다. Group by를 통해 Customer의 ID별로 묶고, 해당 년도, 해당 월에 해당하는 Bill이 있으면 선택하였다. 출력 결과 이름, 주소, 미지급한 금액의 총 합이 나오는데,. 미지급한 금액의 총 합의 경우 Bill의 State를 판별하여 0인 경우(지급X) 해당 Price를 더하고, 1인 경우 0을 더해 미지급한 금액을 계산하였다.

```
------- SELECT QUERY TYPES -------

        1. TYPE I
        2. TYPE II
        3. TYPE III
        4. TYPE IV
        5. TYPE V
        0. QUIT
5
---- TYPE V ----

**Generate the bill for each customer for the past month.**
Which Year? : 2023
Which Month? : 4
-------------------------- Simple Bill --------------------------
--------------Name-----------Address-------------Price--------------
            Taegon         ChunCheon            270000
          Hyeonseo             Wonju                 0
          jinbyeol             Daegu                 0
          Yoonhyuk              Mapo             42000
          Dongheon           Gangnam            132000
-----------------------------------------------------------------
```

+) DataBase에 대한 자료

***Customer**

Customer_ID	Name	Phone_Number	Address	Contract
1	Taegon	01062568591	ChunCheon	1
2	Juhyeon	01012345678	Sillim	0
3	Hyeonseo	01011112222	Wonju	1
4	Seunghwan	01022223333	Sokcho	1
5	Seonghun	01033334444	Gangneung	0
6	Junwoo	01044445555	Suwon	1
7	jinbyeol	01055556666	Daegu	1
8	Chanwoo	01066667777	Ulsan	0
9	Yoonhyuk	01077778888	Mapo	0
10	Dongheon	01088889999	Gangnam	1
NULL	NULL	NULL	NULL	NULL

***Recipient**

Recipient_ID	Name	Phone_Number	Address
1	Ethan	01098765432	Busan
2	Olivia	01045678910	Busan
3	Liam	01023456789	Busan
4	Ava	01067891234	Busan
5	Noah	01054329876	Busan
6	Isabella	01078912345	Busan
7	Mason	01034567890	Busan
8	Sophia	01089104567	Busan
9	Lucas	01067890123	Busan
10	Mia	01093473345	Busan
NULL	NULL	NULL	NULL

***Shipment**

Shipment_ID	Items	Order_Time	Receive_Time	international	Hazardous	Customer_ID	Recipient_ID
1	shoes	2023-04-10 19:00:00	NULL	0	0	10	10
2	Fruit	2023-03-17 08:00:00	2023-03-18 14:00:00	0	0	2	9
3	shirt	2022-12-12 15:00:00	2022-12-18 19:00:00	0	0	3	8
4	snack	2023-05-14 11:00:00	2023-05-15 17:00:00	0	0	4	7
5	laptop	2022-02-02 22:00:00	2022-02-12 12:00:00	0	1	5	6
6	bag	2023-02-12 18:00:00	2023-02-19 07:00:00	1	0	6	5
7	cup	2022-06-02 20:00:00	2022-06-05 09:00:00	0	1	3	4
8	TV	2023-01-30 14:00:00	2023-02-01 10:00:00	0	1	8	3
9	book	2023-05-07 13:00:00	2023-05-17 17:00:00	1	0	2	2
10	keyboard	2023-03-24 12:00:00	2023-03-28 14:00:00	0	0	2	1
11	pants	2023-04-12 13:00:00	NULL	0	0	1	4
12	hat	2023-04-12 16:00:00	NULL	0	0	9	8
13	drink	2023-04-11 20:00:00	2023-04-14 09:00:00	0	0	7	3
14	smartph...	2023-04-13 04:00:00	2023-04-14 10:00:00	0	0	3	9
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

***Bill**

Bill_ID	Pay_Method	Price	Billing_Date	State
1	card	132000	2023-04-12 13:00:00	0
2	kakao	12000	2023-03-17 08:00:00	1
3	phone	32000	2022-12-12 15:00:00	1
4	payco	2000	2023-05-20 11:00:00	0
5	account transfer	2320000	2022-02-02 22:00:00	1
6	card	520000	2023-02-12 18:00:00	1
10	kakao	18000	2022-06-02 20:00:00	1
11	card	2620000	2023-01-30 14:00:00	1
12	payco	42000	2023-05-07 13:00:00	0
13	paypal	200000	2023-03-24 12:00:00	1
14	card	270000	2023-04-12 13:00:00	0
15	card	42000	2023-04-12 13:00:00	0
20	kakao	23000	2023-04-11 20:00:00	1
21	card	1650000	2023-04-13 04:00:00	1
NULL	NULL	NULL	NULL	NULL

*** Service**

Package_Type	Weight	TimeLiness
flat envelope	0.5	longer
flat envelope	1.5	overnight
flat envelope	4	longer
larger box	4	longer
larger box	5	longer
larger box	10	longer
medium box	1	second day
medium box	2	overnight
small box	0.5	overnight
small box	1	overnight
small box	1	second day
small box	1.2	second day
small box	2	second day
NULL	NULL	NULL

***Location**

Shipment_ID	Position_Type	Name_ID	Start_Time	End_Time	State
1	hub	Yongin	2023-04-11 17:00:00	2023-04-14 08:00:00	warehouse
1	truck	1721	2023-04-14 08:00:00	2023-04-14 14:00:00	crash
1	truck	3222	2023-04-10 19:00:00	2023-04-11 17:00:00	shipping
2	hub	Gunpo	2023-03-17 15:00:00	2023-03-18 06:00:00	warehouse
2	truck	2345	2023-03-17 09:00:00	2023-03-17 15:00:00	shipping
2	truck	3456	2023-03-18 06:00:00	2023-03-18 14:00:00	arrive
3	hub	Gonjiam	2022-12-13 06:00:00	2022-12-17 20:00:00	warehouse
3	truck	1721	2022-12-12 19:00:00	2022-12-13 06:00:00	shipping
3	truck	2222	2022-12-17 20:00:00	2022-12-18 19:00:00	arrive
4	hub	Gunpo	2023-05-14 18:00:00	2023-05-15 05:00:00	warehouse
4	truck	4321	2023-05-14 12:00:00	2023-05-14 18:00:00	shipping
4	truck	9999	2023-05-15 05:00:00	2023-05-15 17:00:00	arrive
5	hub	Daejeon	2022-02-08 01:00:00	2022-02-12 06:00:00	warehouse
5	hub	Gonjiam	2022-02-03 21:00:00	2022-02-07 07:00:00	warehouse
5	truck	4830	2022-02-12 06:00:00	2022-02-12 12:00:00	arrive
5	truck	5678	2022-02-07 07:00:00	2022-02-08 01:00:00	shipping
5	truck	8767	2022-02-03 08:00:00	2022-02-03 21:00:00	shipping
6	airplane	172	2023-02-14 02:00:00	2023-02-14 14:00:00	shipping
6	airport	Chicago	2023-02-13 23:00:00	2023-02-14 02:00:00	logistic load
6	Logistic Center	Incheon	2023-02-14 14:00:00	2023-02-18 23:00:00	warehouse
6	truck	4830	2023-02-18 23:00:00	2023-02-19 07:00:00	arrive
6	truck	84730	2023-02-13 09:00:00	2023-02-13 23:00:00	shipping
7	hub	Daejeon	2022-06-03 16:00:00	2022-06-04 22:00:00	warehouse
7	truck	4823	2022-06-03 09:00:00	2022-06-03 16:00:00	shipping
7	truck	7777	2022-06-04 22:00:00	2022-06-05 09:00:00	arrive
8	hub	Yongin	2023-01-30 23:00:00	2023-01-31 22:00:00	warehouse
8	truck	0987	2023-01-31 22:00:00	2023-02-01 10:00:00	arrive
8	truck	3948	2023-01-30 14:00:00	2023-01-30 23:00:00	shipping
9	airplane	341	2023-05-08 09:00:00	2023-05-08 19:00:00	shipping
9	airport	Aberdeen	2023-05-07 21:00:00	2023-05-08 09:00:00	logistic load
9	Logistic Center	Incheon	2023-05-08 19:00:00	2023-05-17 04:00:00	warehouse
9	truck	48732	2023-05-07 15:00:00	2023-05-07 21:00:00	shipping
9	truck	6666	2023-05-17 04:00:00	2023-05-17 17:00:00	arrive
10	hub	Gonjiam	2023-03-25 14:00:00	2023-03-27 23:00:00	warehouse
10	truck	3821	2023-03-27 23:00:00	2023-03-28 14:00:00	arrive
10	truck	4932	2023-03-25 07:00:00	2023-03-25 14:00:00	shipping
11	hub	Yongin	2023-04-12 20:00:00	2023-04-14 08:00:00	warehouse
11	truck	1234	2023-04-12 14:00:00	2023-04-12 20:00:00	shipping
11	truck	1721	2023-04-14 08:00:00	2023-04-14 14:00:00	crash
12	hub	Yongin	2023-04-13 10:00:00	2023-04-14 08:00:00	warehouse
12	truck	1234	2023-04-12 19:00:00	2023-04-13 10:00:00	shipping
12	truck	1721	2023-04-14 08:00:00	2023-04-14 14:00:00	crash
13	hub	Yongin	2023-04-12 18:00:00	2023-04-14 08:00:00	warehouse
13	truck	1721	2023-04-14 08:00:00	2023-04-14 09:00:00	arrive
13	truck	2342	2023-04-11 21:00:00	2023-04-12 18:00:00	shipping
14	hub	Yongin	2023-04-13 14:00:00	2023-04-14 08:00:00	warehouse
14	truck	1423	2023-04-13 06:00:00	2023-04-13 14:00:00	shipping
14	truck	1721	2023-04-14 08:00:00	2023-04-14 10:00:00	arrive
NULL	NULL	NULL	NULL	NULL	NULL

***Ship_Ser**

Shipment_ID	Package_Type	Weight	Timeliness
3	flat envelope	0.5	longer
11	flat envelope	1.5	overnight
9	flat envelope	4	longer
5	larger box	4	longer
13	larger box	5	longer
8	larger box	10	longer
10	medium box	1	second day
1	medium box	2	overnight
6	medium box	2	overnight
12	small box	0.5	overnight
7	small box	1	overnight
2	small box	1	second day
4	small box	1.2	second day
14	small box	2	second day
NULL	NULL	NULL	NULL

***Ship_Bill**

Shipment_ID	Bill_ID
1	1
2	2
3	3
4	4
5	5
6	6
7	10
8	11
9	12
10	13
11	14
12	15
13	20
14	21
NULL	NULL

***Bill_Cus**

Bill_ID	Customer_ID
14	1
2	2
12	2
13	2
3	3
10	3
21	3
4	4
5	5
6	6
20	7
11	8
15	9
1	10
NULL	NULL