데이터 베이스 Project2

20191662 최찬우

<BCNF Decomposition>-Entity

1. Customer(ID, name, address, phone_num,payment_method)

Functional Dependency

1. ID->name,address,phone_num, payment_method

고객의 ID 속성의 값 각각에 대해 항상 고객의 이름, 주소, 전화번호, 지불 방법의 값이 오직 하나만 연관되어 있기 때문이다.

ID->name,address,phone_num,payment_method는

nontrivial한 functional dependency이므로

(ID)+가 Customer의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = ID,name,address,phone_num,payment_method (by 1)

이므로

Customer Entity는 BCNF이다.

2. Shipment(ID,date,status,cost)

Functional Dependency

1. ID->date,cost,status

배송 번호(ID) 속성의 값 각각에 대해 항상 배송 날짜(date), 배송 비용(cost), 배송 상태(status)의 값이 오직 하나만 연관되어 있기 때문이다.

ID->date,cost,status는 nontrivial한 functional dependency이므로

(ID)+가 Shipment의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = date,cost,status (by 1)

이므로

Shipment Entity는 BCNF이다.

3. Receipient(ID,name,address,phone_num)

Functional Dependency

1. ID->name,address,phone_num, payment_method

수신자의 ID 속성의 값 각각에 대해 항상 수신자의 이름(name), 주소(address), 전화번호(phone_num)의 값이 오직 하나만 연관되어 있기 때문이다.

ID->name,address,phone_num는 nontrivial한 functional dependency이므로

(ID)+가 Receipient의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = ID,name,address,phone_num (by 1)

이므로

Receipient Entity는 BCNF이다.

4. Vehicle(Vehicle_number,type)

Functional Dependency

1. Vehicle_number-> type

차량에 부여된 고유 번호(Vehicle_number)속성의 값 각각에 대해 항상 차량 종류(type)의 값이 오직 하나만 연관되어 있기 때문이다.

Vehicle_number-> type은 nontrivial한 functional dependency이므로

(Vehicle_number)+가 Vehicle의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Vehicle_number)+ = Vehicle_number,type (by 1)

이므로

Vehicle Entity는 BCNF이다.

5. Service_type(ID,Package Type,Weight,Timeliness)

Functional Dependency

1. ID-> Package Type,Weight,Timeliness

패키지의 여러 정보에 따라 이미 정해놓은 서비스 타입의 고유 번호(ID)속성의 값 각각에 대해 항상 포장지의 종류(Pacakge Type), 배송품의 무게(Weight), 배송 기한(Timeliness)의 값이 오직 하나만 연관되어 있기 때문이다

ID-> Package Type,Weight,Timeliness는 nontrivial한 functional dependency이므로

(ID)+가 Service_type의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(ID)+ = Package Type,Weight,Timeliness (by 1)

이므로

Service_type은 BCNF이다.

6. Package_Info(Shipment ID, Package name, Description)

Functional Dependency

1. Shipment ID,Description -> Package name

Project1에서 작성한 ER-Model에 따르면, 각각의 배송에 대해 package의 정보를 저장해야하고 Shipment와 Package Info가 one to many이므로 다른 배송에 대해서는 다른 물건이 배송된다는 가정이 있었다. 이에 따르면 배송 번호(Shipment ID)와 배송 상품에 대한 설명(Description) 속성의

값 각각에 대해 항상 배송 물품의 이름(Package name)의 값이 오직 하나만 연관되어 있다.

이러한 가정하에 Shipment ID,Description -> Package name는 nontrivial한 functional dependency 이므로 (Shipment ID,Description)+가 Package Info의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID,Description)+ = Package name,Description,Shipment ID

이므로

Package Info는 BCNF이다.

<BCNF Decomposition>-Relationship

1. Cus_Ship(Shipment ID, Customer ID)

Functional Dependency

1.Shipment ID->Customer ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 고객의 ID(Customer ID) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Customer ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Cus_Ship의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Customer ID (by 1)

이므로

Cus_ship은 BCNF이다.

2. Ship_Service(Shipment ID, Service_type ID)

Functional Dependency

1. Shipment ID->Service_type ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 서비스 타입의 고유 번호(ID)속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Service_type ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Ship_Service의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Service_type ID (by 1)

이므로

Ship_Service는 BCNF이다.

3. Rec_Ship(Shipment ID, Receipient ID)

Functional Dependency

1. Shipment ID->Receipient ID

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 수신자의 ID(Receipient ID) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID->Receipient ID는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Rec_Ship의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Receipient ID (by 1)

이므로

Rec_ship은 BCNF이다.

4. Delivery_Vehicle(Shipment ID, Vehicle number)

Functional Dependency

1. Shipment ID-> Vehicle number

배송 번호(Shipment ID) 속성의 값 각각에 대해 항상 고유한 차량 번호(Vehicle number) 속성의 값이 오직 하나만 연관되어 있기 때문이다.

Shipment ID-> Vehicle number는 nontrivial한 Functional Dependency이므로

(Shipment ID)+가 Delivery_Vehicle의 속성을 전부 가지는지 판단하면 된다.

자명하게도

(Shipment ID)+ = Shipment ID, Vehicle number (by 1)

이므로

Delivery_Vehicle은 BCNF이다.

5. Hazardous/Imternational(Shipment ID, Package name, Description)

Functional Dependency

1. Package name,Description -> Shipment ID

앞에서 Package Info에서 언급했던 것처럼 배송 번호(Shipment ID)와 배송 상품에 대한 설명(Description) 속성의 값 각각에 대해 항상 배송 물품의 이름(Package name)의 값이 오직 하나만 연관되어 있다.

이러한 가정하에 Shipment ID,Description -> Package name는 nontrivial한 functional dependency 이므로 (Shipment ID,Description)+가 Package Info의 속성을 전부 가지는지 판단하면 된다.

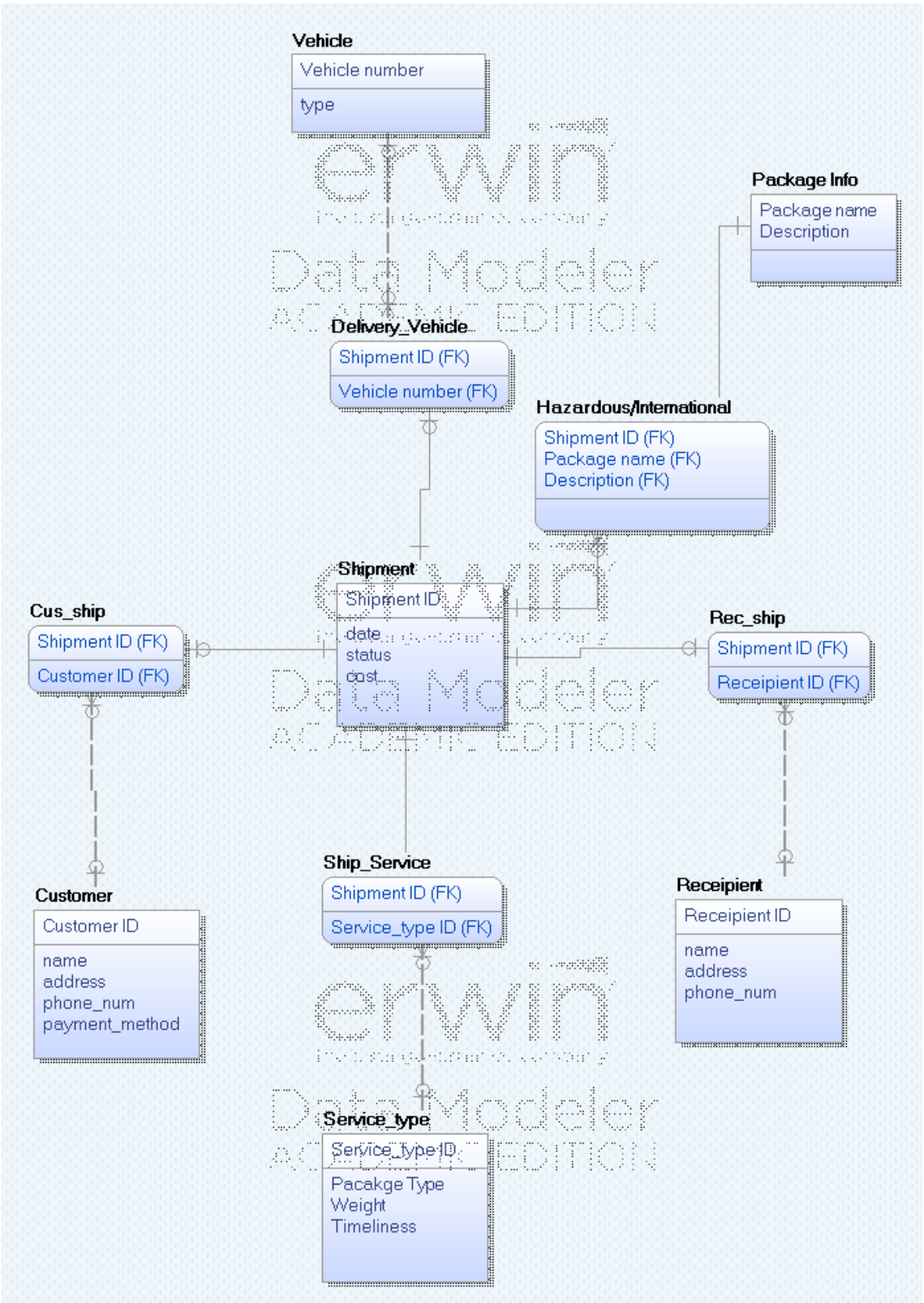
자명하게도

(Shipment ID,Description)+ = Package name,Description,Shipment ID

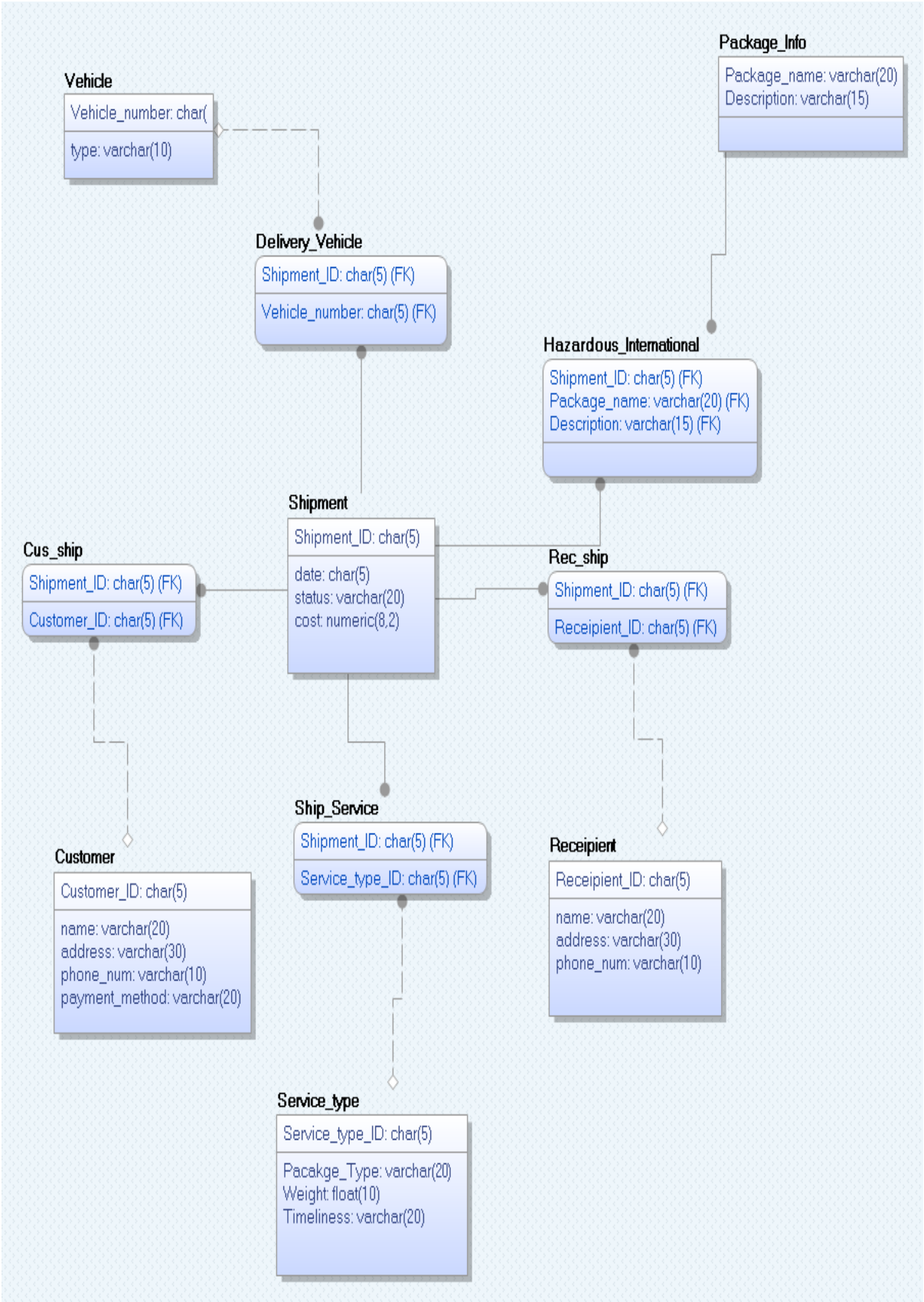
이므로

Hazardous/Imternational는 BCNF이다.

이를 바탕으로 logical Schema Diagram을 그리면 다음과 같으며 PRJ1에서와 동일하다.



<Physical Diagram & OBDC Implementation>



1. Customer

```
create table Customer
(Customer_ID    char(5),
name    varchar(20),
address varchar(30),
phone_num   varchar(10),
payment_method  varchar(20),
primary key(Customer_ID)
);
```

ID: 고객이 회사의 서비스를 이용한 순간 부여되는 고유 번호(primary key)

-5자리로 고정된 ID이기 때문에 char type으로 선언

name: 고객의 이름

-이름의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

address: 고객의 주소

-주소의 최대 글자만 20자로 정해주고 여러 주소의 길이를 수용할 수 있게 varchar type으로 선언

phone_num: 고객의 핸드폰 번호

-배달의 민족처럼 고객 보호 번호를 위해 랜덤한 번호를 최대 10글자로 정의하기 위해 varchar type으로 선언

payment_method: 지불 방식에 대한 속성(account number 또는 credit card가 들어감)

-지불 방법은 acccount number 또는 credit card라고 적히기 때문에 varchar type으로 선언

2. Shipment

```
create table Shipment(
Shipment_ID char(5),
date    char(8),
status  varchar(20),
cost    numeric(8,2),
primary key(Shipment_ID)
);
```

ID: 배송이 되는 순간 부여되는 고유한 배송 번호(primary key)

-5자리로 고정된 ID이기 때문에 char type으로 선언

date: 배송 날짜

-23년 1월 1일이면 20230101의 형태로 저장하기 위해 char(8)로 선언

status: 배송 상태(배송 중, 배송 완료(제시간), 배송 완료(지연) 등의 정보가 들어감)

-배송 상태에 대한 여러 정보가 들어가기 때문에 varchar type으로 선언

cost: 현재 배송에 지불되는 비용

- 비용에 대한 정보이므로 numeric type으로 선언

3. Receipient

```
create table Receipient(
Receipient_ID   char(5),
name    varchar(20),
address varchar(30),
phone_num   varchar(10),
primary key(Receipient_ID)
);
```

ID: 고객이 특정 수신자에게 배송하겠다고 서비스를 이용한 순간 수신자에게 고유 번호가 부여됨 (primary key)

-5자리로 고정된 ID이기 때문에 char type으로 선언

name: 수신자의 이름

-이름의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

address: 수신자의 주소

-주소의 최대 글자만 20자로 정해주고 여러 주소의 길이를 수용할 수 있게 varchar type으로 선언

phone_num: 수신자의 휴대폰 번호

-배달의 민족처럼 고객 보호 번호를 위해 랜덤한 번호를 최대 10글자로 정의하기 위해 varchar type으로 선언

4. Service_type

```
create table Service_type(
Service_type_ID char(5),
Package_Type    varchar(20),
Weight  float(10),
Timeliness  varchar(20),
primary key(Service_type_ID)
);
```

ID: 패키지의 여러 정보에 따라 이미 정해놓은 서비스 타입의 고유 번호(primary key)

-5자리로 고정된 ID이기 때문에 char type으로 선언

Package Type: 포장지의 종류

-포장 타입의 최대 글자만 20자로 정해주고 여러 이름의 길이를 수용할 수 있게 varchar type으로 선언

Weight: 배송품의 무게

-무게의 tpye은 float type으로 선언하여 디테일한 측정이 가능하게 만듦

Timeliness: 배송 기한

-배송 기한은 overnight, second_day 등의 정보가 들어가므로 varchar type으로 선언

5. Vehicle

```
create table Vehicle(
Vehicle_number  char(5) ,
type    varchar(10)  ,
primary key(Vehicle_number)
);
```

Vehicle_number: 각 차량의 차량 번호(primary key)

-차량의 번호는 회사에서 고유하게 지정해주며 5자리까지 가능하여 char type으로 선언

type: 차량 종류(truck, plane 등)

-차량 종류는 다양한 이름 길이를 가질 수 있으므로 varchar type으로 선언

6. Package_Info

```
create table Package_Info(
Package_name varchar(20)  ,
Description varchar(15)  ,
Shipment_ID varchar(10)  ,
primary key(Package_name,Description),
foreign key (Shipment_ID) references Shipment (Shipment_ID)
);
```

Package name: 배송 물품의 이름

-배송 물품의 이름은 다양할 수 있으므로 varchar type으로 선언

Description: 배송 물품의 구체적 정보

-구체적 정보 또한 국제배송, 위험 물질 등의 정보등이 적혀있어 그 길이가 다양할 수 있으므로 varchar type으로 선언

7. Cus_ship

```
create table Cus_ship(
Customer_ID char(5)  ,
Shipment_ID char(5)  ,
primary key(Shipment_ID),
foreign key(Shipment_ID) references Shipment(Shipment_ID),
foreign key(Customer_ID) references Customer(Customer_ID)
);
```

Cus_ship은 Customer과 Shipment 간의 one to may relationship이므로 many 쪽의 Shipment의 primary key인 Shipment_ID가 Cus_ship의 primary key로 정의되고, Customer과 Shipment의 primary key는 Cus_ship의 foreign key로 정의된다.

8. Rec_ship

```
create table Rec_ship(
Receipient_ID   char(5)  ,
Shipment_ID char(5)  ,
primary key(Shipment_ID),
foreign key(Shipment_ID) references Shipment(Shipment_ID),
foreign key(Receipient_ID) references Receipient(Receipient_ID)
);
```

Rec_ship은 Receipient와 hipment 간의 one to may relationship이므로 many 쪽의 Shipment의 primary key인 Shipment_ID가 Rec_ship의 primary key로 정의되고, Receipient와 Shipment의 primary key는 Cus_ship의 foreign key로 정의된다.

.

9. Ship_Service

```
create table Ship_Service(
Service_type_ID     char(5)  ,
Shipment_ID char(5)  ,
primary key(Shipment_ID),
foreign key(Shipment_ID) references Shipment(Shipment_ID),
foreign key(Service_type_ID) references Service_type(Service_type_ID)
);
```

Ship_Service는 Shipment와 Service_type의 many to one relationship이므로 many쪽의 Shipment의 primary key인 Shipment_ID가 Ship_Service의 primary key로 정의되고, Shipment와 Ship_Service의 primary key는 Ship_Service의 foreign key로 정의된다.

10. Delivery_Vehicle

```
create table Delivery_Vehicle(
Vehicle_number  char(5)  ,
Shipment_ID char(5)  ,
primary key(Shipment_ID),
foreign key(Shipment_ID) references Shipment(Shipment_ID),
foreign key(Vehicle_number) references Vehicle(Vehicle_number)
);
```

Delivery_Vehicle은 Shipment와 Vehicle의 many to one 관계이므로 many 쪽의 Shipment의 primary key인 Shipment_ID가 Delivery_Vehicle의 priamary key로 정의되고, Shipment와 Vehicle의 primary key는 Delivery_Vehicle의 foreign key로 정의된다.

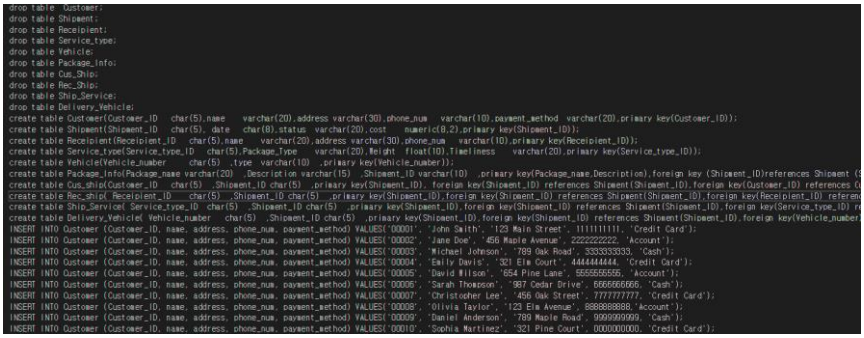
<OBDC C language codes>

1. MySQL 서버에 database file update

```
FILE* fp = fopen("dbfiletxt.txt", "r");
while (!feof(fp)) {
    fgets(buf, MAXLINE, fp);
    mysql_query(connection, buf);
}

fclose(fp);
```

-> dbfiletxt.txt를 r 모드로 읽어와서 업로드한다.



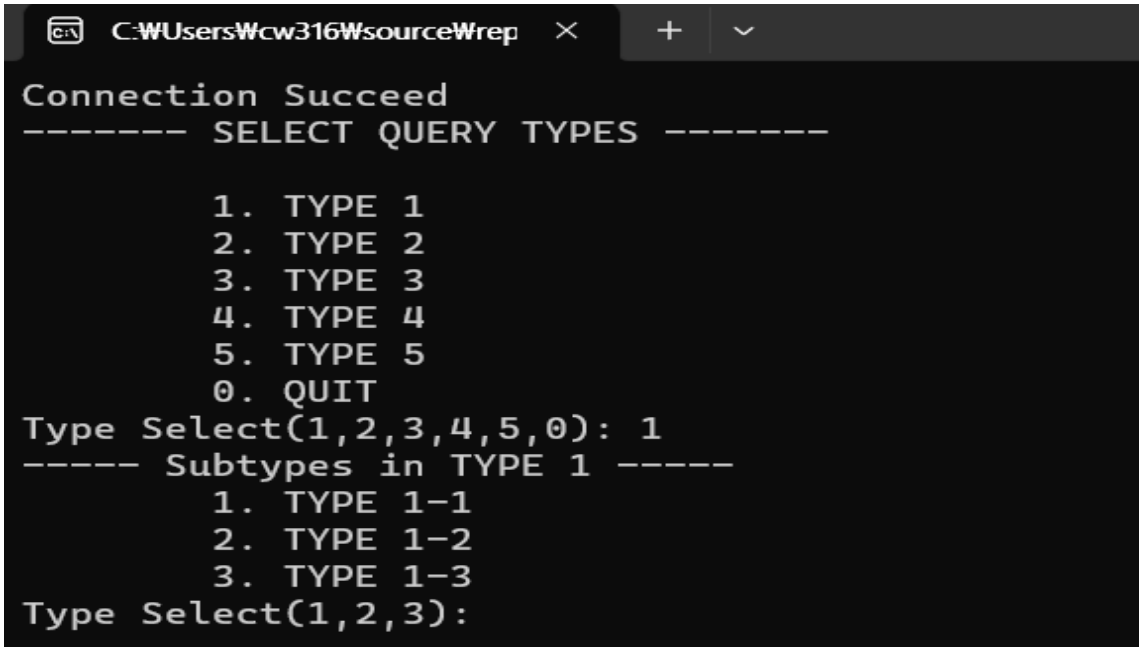
```
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11003', '20230603', 'In Transit', 60.25);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11004', '20230604', 'Delivered(on time)', 85.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11005', '20230605', 'In Transit', 70.50);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11006', '20230606', 'Delivered(on time)', 65.25);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11008', '20230608', 'Delivered(on time)', 55.50);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11009', '20230609', 'In Transit', 90.25);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11024', '20230510', 'Delivered(delayed)', 10.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11025', '20230510', 'Delivered(delayed)', 11.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11026', '20230510', 'Delivered(delayed)', 12.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11027', '20230510', 'Delivered(delayed)', 12.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11028', '20230410', 'Delivered(delayed)', 13.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11029', '20230410', 'Delivered(delayed)', 14.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11030', '20230410', 'Delivered(delayed)', 15.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11031', '20230410', 'Delivered(delayed)', 16.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11032', '20230310', 'Delivered(delayed)', 17.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11033', '20230310', 'Delivered(delayed)', 18.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11034', '20230310', 'Delivered(delayed)', 19.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11035', '20230310', 'Delivered(delayed)', 29.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11036', '20230210', 'Delivered(delayed)', 20.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11037', '20230210', 'Delivered(delayed)', 14.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11038', '20230210', 'Delivered(delayed)', 16.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11039', '20230210', 'Delivered(delayed)', 93.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11040', '20230110', 'Delivered(delayed)', 54.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11041', '20230110', 'Delivered(delayed)', 42.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11042', '20230110', 'Delivered(delayed)', 62.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11043', '20230110', 'Delivered(delayed)', 39.75);
INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11010', '20230410', 'Delivered(delayed)', 95.75);
```

(dbfiletxt.txt 의 일부분 예시)

2. menu 창 구현

```c
while (1) {
    printf("------- SELECT QUERY TYPES -------₩n₩n");
    printf("₩t1. TYPE 1₩n");
    printf("₩t2. TYPE 2₩n");
    printf("₩t3. TYPE 3₩n");
    printf("₩t4. TYPE 4₩n");
    printf("₩t5. TYPE 5₩n");
    printf("₩t0. QUIT₩n");
    printf("Type Select(1,2,3,4,5,0): ");
    scanf("%d", &num);
    start = 1;
    int year = 0;
    char s1[10];
    memset(s1, 0, sizeof(char) * 10);
    memset(query, 0, sizeof(char)*MAXLINE);
    switch (num) {
    case 1:
        printf("----- Subtypes in TYPE 1 -----₩n");
        printf("₩t1. TYPE 1-1₩n");
        printf("₩t2. TYPE 1-2₩n");
        printf("₩t3. TYPE 1-3₩n");
        printf("Type Select(1,2,3): ");
        scanf("%d", &k);
        if (k == 1) {
            printf("---- TYPE 1-1 ----₩n₩n");
            printf("**Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash**₩n");
```

->code



->실행 결과

3. 각 type별 SQL 구문, 코드, 실행결과

(Type 1-1)

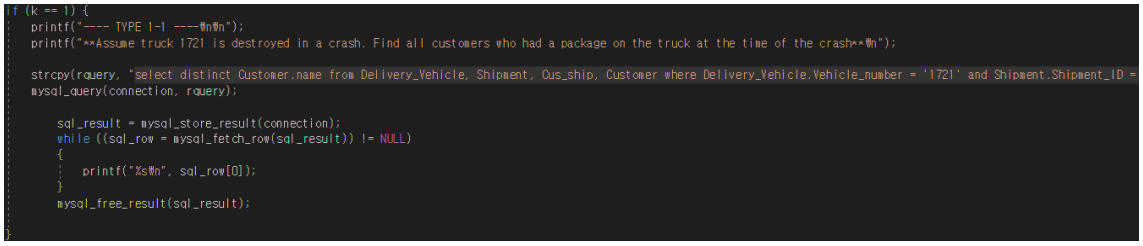
Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

"Delivery_Vehicle set에서 Vehicle_number(1721번)을 조회하면 해당 트럭의 배송 운행 이력으로도 볼 수 있는 Shipment ID들을 알 수 있고, 여기서 얻은 모든 Shipment ID들 중에 현재 배송 중인 status의 Shipment ID를 찾는다면, 고객의 정보를 찾을 수 있을 것이다"

```
select distinct Customer.name
from Delivery_Vehicle, Shipment, Cus_ship, Customer
where Delivery_Vehicle.Vehicle_number = '1721'
 and Shipment.Shipment_ID = Delivery_Vehicle.Shipment_ID
 and Shipment.status = 'In Transit'
 and Cus_ship.Shipment_ID = Shipment.Shipment_ID
 and Cus_Ship.Customer_ID = Customer.Customer_ID
```

->실제 구현

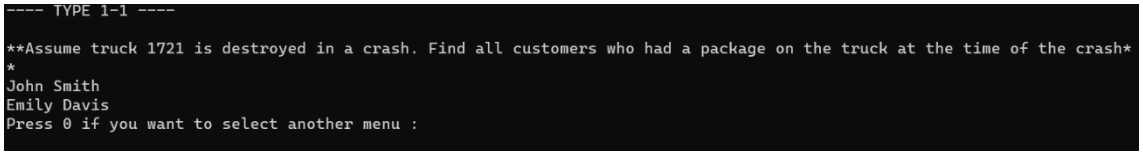


```
if (k == 1) {
    printf("---- TYPE 1-1 ----\n\n");
    printf("**Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash**\n");

    strcpy(query, "select distinct Customer.name from Delivery_Vehicle, Shipment, Cus_ship, Customer where Delivery_Vehicle.Vehicle_number = '1721' and Shipment.Shipment_ID =
    mysql_query(connection, query);

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

->code



```
---- TYPE 1-1 ----

**Assume truck 1721 is destroyed in a crash. Find all customers who had a package on the truck at the time of the crash*
*
John Smith
Emily Davis
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11001','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11002','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11003','10102');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11005','10103');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11007','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11009','10105');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11013','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11014','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment_ID는 “11001, 11002, 11007, 11013, 11014”

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 운행중인 Shipment_ID는 "11001, 11007"

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11001','00001');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11007','00004');

-> 이와 연관된 Customer_ID는 "00001, 00004"

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00001', 'John Smith', '123 Main Street', 1111111111, 'Credit Card');

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00004', 'Emily Davis', '321 Elm Court', 4444444444, 'Credit Card');

->00001과 00004의 ID를 가지는 Customer의 이름은 "John Smith, Emily Davis"

따라서 실행 결과가 맞다.

(Type 1-2) -Type 1-1과 접근 방법 거의 동일

Assume truck 1721 is destroyed in a crash. Find all recipients who had a package on that truck at the time of the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

"Delivery_Vehicle set에서 Vehicle_number(1721번)을 조회하면 해당 트럭의 배송 운행 이력으로도 볼 수 있는 Shipment ID들을 알 수 있고, 여기서 얻은 모든 Shipment ID들 중에 현재 배송 중인 status의 Shipment ID를 찾는다면, 수신자의 정보를 찾을 수 있을 것이다"

```
select distinct Receipient.name
from Delivery_Vehicle, Shipment, Rec_ship, Receipient
where Delivery_Vehicle.Vehicle_number = '1721'
and Shipment.Shipment_ID = Delivery_Vehicle.Shipment_ID
and Shipment.status = 'In Transit'
and Rec_ship.Shipment_ID = Shipment.Shipment_ID
and Rec_Ship.Receipient_ID = Receipient.Receipient_ID
```

->실제 구현

```
else if (k == 2) {
    printf("---- TYPE 1-2 ----₩n₩n");
    printf("**Assume truck 1721 is destroyed in a crash. Find all recipients who had a package on that truck at the time of the
    strcpy(rquery, "select distinct Receipient.name from Delivery_Vehicle, Shipment, Rec_ship, Receipient where Delivery_Vehicl
    mysql_query(connection, rquery);

        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%s₩n", sql_row[0]);
        }
        mysql_free_result(sql_result);
}
```

->code

```
---- TYPE 1-2 ----

**Assume truck 1721 is destroyed in a crash. Find all recipients who had a package on that truck at the time of the crash**
Alex Johnson
Ava Thompson
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11001','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11002','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11003','10102');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11005','10103');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11007','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11009','10105');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11013','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11014','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment_ID는 "11001, 11002, 11007, 11013, 11014"

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 운행중인 Shipment_ID는 "11001, 11007"

INSERT INTO Rec_ship(Shipment_ID, Receipient_ID) VALUES('11001','22001');

INSERT INTO Rec_ship(Shipment_ID, Receipient_ID) VALUES('11007','22004');

-> 이와 연관된 Receipient_ID는 "22001, 22004"

INSERT INTO Receipient (Receipient_ID, name, address, phone_num) VALUES('22001', 'Alex Johnson', '567 Park Avenue', 1234567890);

INSERT INTO Receipient (Receipient_ID, name, address, phone_num) VALUES('22004', 'Ava Thompson', '1098 Vineyard Drive', 9998887777);

->22001과 22004의 ID를 가지는 Receipient의 이름은 "Alex Johnson, Ava Thompson"이다.

따라서 실행 결과가 맞다.

(Type 1-3)

Assume truck 1721 is destroyed in a crash. Find the last successful delivery by that truck prior to the crash

prj1에서 설명하였던 방법(prj1 보고서 참고)

Delivery_Vehicle set에서 Vehicle_number(1721번)을 조회하고 여기서 얻은 모든 Shipment ID들을 이용해 가장 최근 date의 Shipment ID를 찾는다면 관련 정보를 얻을 수 있다.

```
select max(Shipment.date)
from Delivery_Vehicle, Shipment
where Delivery_Vehicle.Vehicle_number = '1721'
and Shipment.Shipment_ID = Delivery_Vehicle.Shipment_ID
and (Shipment.status = 'Delivered(on time)' or Shipment.status = 'Delivered(delayed)')
```

->실제 구현

```
else if (k == 3) {
    printf("**Assume truck 1721 is destroyed in a crash. Find the last successful delivery by that truck prior to the crash**₩n");

    strcpy(rquery, "select max(Shipment.date) from Delivery_Vehicle, Shipment where Delivery_Vehicle.Vehicle_number = '1721' and Shi
     mysql_query(connection, rquery);

        sql_result = mysql_store_result(connection);
        while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
        {
            printf("%s₩n", sql_row[0]);
        }
        mysql_free_result(sql_result);

}
```

->code

```
**Assume truck 1721 is destroyed in a crash. Find the last successful delivery by that truck prior to the crash**
20230601
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11001','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11002','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11003','10102');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11005','10103');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11007','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11009','10105');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11013','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11014','1721');

INSERT INTO Delivery_Vehicle(Shipment_ID, Vehicle_number) VALUES('11004',10107');

->여기서 1721번 트럭과 연관되어 있는 Shipment_ID는 "11001, 11002, 11007, 11013, 11014"

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11013', '20230315', 'Delivered(on time)', 50.24);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->여기서 Delivered된 Shipment_ID는 "11002, 11013, 11014"

여기서 max(Shipment.date)는

23년6월1일인 20230601이므로 출력결과가 옳게 나오는 것을 알 수 있다.

(Type 2)

Find the customer who has shipped the most packages in the past year

prj1에서 설명하였던 방법(prj1 보고서 참고)

Cus_ship set에는 Customer의 ID와 Shipment ID(primary key)가 저장되어 있다. 따라서 Cus_ship

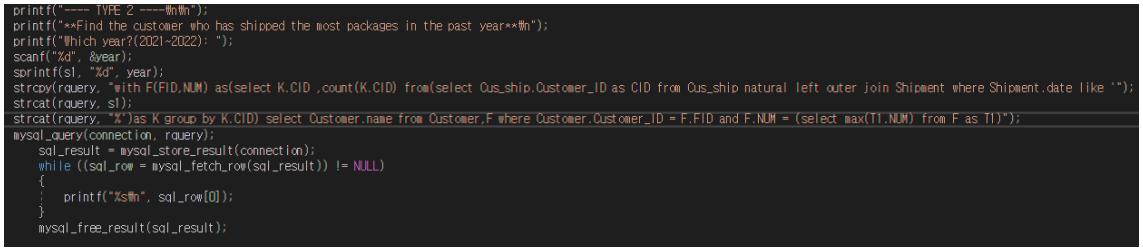
set에서 각 Customer ID별 Shipment ID들을 조회하고, 해당 Shipment ID를 이용해 Shipment set에서 date가 지정된 과거 년도인 것만 찾은 다음 이를 모든 고객에 대해 반복하여 가장 많은 package를 배송한 고객의 ID를 찾을 수 있다.

```
with F(FID,NUM) as(
select K.CID ,count(K.CID)
from(select Cus_ship.Customer_ID as CID
    from Cus_ship natural left outer join Shipment
    where Shipment.date like '2022%'
    )as K
group by K.CID
)
select Customer.name
from Customer,F
where Customer.Customer_ID = F.FID and F.NUM = (select max(T1.NUM) from F as T1)
```

->실제 구현 내용

with 절에서 임시로 만든 relation F의 attribute는 FID와 NUM으로 이것이 의미하는 것은 특정 년도(위에서는 2022년)에 배송한 Customer의 ID와 몇 번 배송했는지를 나타낸다.

이를 통해 결국 알고 싶은 것은 고객의 이름이므로, Customer와 F를 join하여 Customer의 ID와 F의 ID가 같은 것을 찾으며, F의 NUM중 가장 큰 값을 찾으면 Customer의 이름을 알 수 있다.



```
printf("---- TYPE 2 ----\n\n");
printf("**Find the customer who has shipped the most packages in the past year**\n");
printf("Which year?(2021~2022): ");
scanf("%d", &year);
sprintf(sl, "%d", year);
strcpy(rquery, "with F(FID,NUM) as(select K.CID ,count(K.CID) from(select Cus_ship.Customer_ID as CID from Cus_ship natural left outer join Shipment where Shipment.date like '");
strcat(rquery, sl);
strcat(rquery, "%')as K group by K.CID) select Customer.name from Customer,F where Customer.Customer_ID = F.FID and F.NUM = (select max(T1.NUM) from F as T1)");
mysql_query(connection, rquery);
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%s\n", sql_row[0]);
    }
    mysql_free_result(sql_result);
```

->code

프로그램에서는 연도를 입력받아야 특정 연도를 확인할 수 있으므로 with~ like ' 까지를 먼저 rquery에 집어넣고, 입력받은 연도의 타입을 sprintf을 이용해 문자열로 바꿔준 뒤 rquery에 strcat을 이용해 붙여주고, 다시 %'~끝까지 rquery에 붙여주어서 SQL 구문을 완성시키는 방법을 선택하였다.

```
**Find the customer who has shipped the most packages in the past year**
Which year?(2021~2022): 2021
David Wilson
Press 0 if you want to select another menu : 0
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
Type Select(1,2,3,4,5,0): 2
---- TYPE 2 ----

**Find the customer who has shipped the most packages in the past year**
Which year?(2021~2022): 2022
Emily Davis
David Wilson
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11016', '20220117', 'Delivered(on time)', 300.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11017', '20220118', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11018', '20220119', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11019', '20220120', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11020', '20220121', 'Delivered(on time)', 90.19);

->2022년도에 대한 Shipment_ID

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11016','00003');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11017','00004');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11018','00004');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11019','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11020','00005');

->해당 Shipment_ID와 연관된 Customer_ID, 00004와 00005가 각 2번씩 가장 많이 배송을 한 것을 알 수 있다.

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00004', 'Emily Davis', '321 Elm Court', 4444444444, 'Credit Card');

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00005', 'David Wilson', '654 Pine Lane', 5555555555, 'Account');

->이와 연관된 Customer의 이름은 "Emily Davis, David Wilson"이며 2022년도에 대해서 맞다

마찬가지로

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11021', '20210122', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11022', '20210123', 'Delivered(on time)', 37.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11023', '20210124', 'Delivered(on time)', 67.19);

->2021년 Shipment_ID를 찾고

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11021','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11022','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11023','00006');

->Customer의 ID를 찾으면 00005가 두 번으로 가장 많음을 알 수 있고

->00005의 이름은 앞에서 봤듯이 David Wilson이다.

따라서 실행 결과는 맞다

(Type 3)

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

prj1에서 설명하였던 방법(prj1 보고서 참고)

Cus_ship set에서 각 Customer ID별 Shipment ID들을 조회하고, 해당 Shipment ID를 이용해 Shipment set에서 date가 작년인것만 찾은 다음 그것들의 cost를 전부 더한다. 그리고 이를 모든 고객에 대해 반복하여 가장 비용이 큰 고객을 찾는다.

```
with F(FID,NUM) as(
select K.CID ,sum(K.C_COST)
from(select Cus_ship.Customer_ID as CID, Shipment.cost as C_COST
    from Cus_ship natural left outer join Shipment
    where Shipment.date like '2022%')as K
group by K.CID
)
select Customer.name
from Customer,F
where Customer.Customer_ID = F.FID and F.NUM = (select max(T1.NUM) from F as T1)
```

->실제 구현 내용

2번과 거의 동일하게 with 절에서 임시로 만든 relation F의 attribute는 FID와 NUM으로 이것이 의미하는 것은 특정 년도(위에서는 2022년)에 배송한 Customer의 ID와 해당 Customer가 배송에 소비한 비용의 합을 나타낸다.

이를 통해 결국 알고 싶은 것은 고객의 이름이므로, Customer와 F를 join하여 Customer의 ID와 F의 ID가 같은 것을 찾으며, F의 NUM중 가장 큰 값을 찾으면 Customer의 이름을 알 수 있다.

```
case 3:
    printf("---- TYPE 3 ----\n\n");
    printf("**Find the customer who has spent the most money on shipping in the past year**\n");
    printf("Which year?(2021~2022): ");
    scanf("%d", &year);
    sprintf(s1, "%d", year);
    strcpy(rquery, "with F(FID,NUM) as(select K.CID ,sum(K.C_COST)from(select Cus_ship.Customer_ID as CID, Shipment.cost as C_COST from Cus_ship natural left outer join Shipment where Shipment.date like '");
    strcat(rquery, s1);
    strcat(rquery, "%')as K group by K.CID) select Customer.name from Customer,F where Customer.Customer_ID = F.FID and F.NUM = (select max(T1.NUM) from F as T1)");
    mysql_query(connection, rquery);
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%s\n", sql_row[0]);
    }
    mysql_free_result(sql_result);
```

->code

sql query를 만드는 방식은 Type2와 완벽히 동일하여 설명을 생략함

```
**Find the customer who has spent the most money on shipping in the past year**
Which year?(2021~2022): 2022
Michael Johnson
Press 0 if you want to select another menu : 0
------- SELECT QUERY TYPES -------

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
Type Select(1,2,3,4,5,0): 3
---- TYPE 3 ----

**Find the customer who has spent the most money on shipping in the past year**
Which year?(2021~2022): 2021
David Wilson
Press 0 if you want to select another menu :
```

->실행결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11016', '20220117', 'Delivered(on time)',

300.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11017', '20220118', 'Delivered(on time)',

67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11018', '20220119', 'Delivered(on time)',

67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11019', '20220120', 'Delivered(on time)',

67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11020', '20220121', 'Delivered(on time)',

90.19);

->2022년도에 대한 Shipment_ID

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11016','00003');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11017','00004');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11018','00004');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11019','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11020','00005');

->이를 통해 만들어지는 F는

FID	NUM
00003	300.19
00004	134.38
00005	157.38

따라서 가장 많은 비용을 낸 고객 ID는 00003

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00003', 'Michael Johnson', '789 Oak Road', 3333333333, 'Cash');

->그는 바로 Michael Johnson이며 실행 결과와 일치한다.

마찬가지로

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11021', '20210122', 'Delivered(on time)', 67.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11022', '20210123', 'Delivered(on time)', 37.19);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11023', '20210124', 'Delivered(on time)',

67.19);

->2021년 Shipment_ID를 찾고

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11021','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11022','00005');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11023','00006');

->Customer의 ID를 찾으면 F는

FID	NUM
00005	104.38
00006	67.19

이므로 00005라는 ID를 가지는 고객이 가장 많이 소비하였으며

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00005',

'David Wilson', '654 Pine Lane', 5555555555, 'Account');

->이는 바로 David Wilson이다

따라서 실행 결과가 맞다

(Type 4)

Find those packages that were not delivered within the promised time

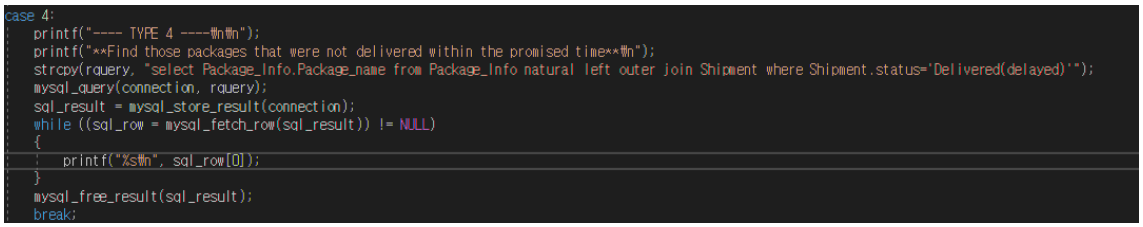
prj1에서 설명하였던 방법(prj1 보고서 참고)

Shipment Set에서 status가 배송 완료(지연)과 관련된 정보를 찾으면 된다.

```
select Package_Info.Package_name
from Package_Info natural left outer join Shipment
where Shipment.status='Delivered(delayed)'
```

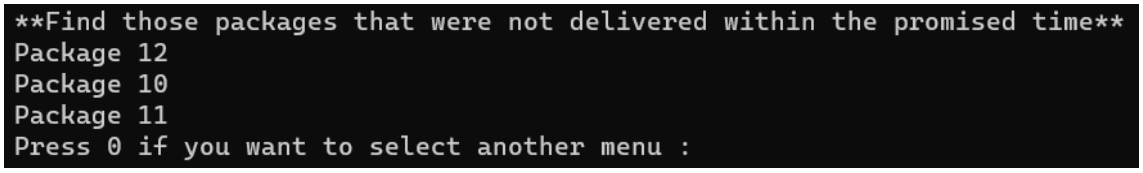
->실제 구현 내용

Package_Info natural join left outer join을 하면 Package_Info의 Shipment_ID을 기준으로 Shipment과 일치하는 행들만의 테이블이 형성된다. 그러한 테이블에서 배송완료(지연)이라는 조건을 만족하는 행을 찾으면 그것에 해당하는 Package의 이름을 알 수 있다.



```
case 4:
    printf("---- TYPE 4 ----₩n₩n");
    printf("**Find those packages that were not delivered within the promised time**₩n");
    strcpy(rquery, "select Package_Info.Package_name from Package_Info natural left outer join Shipment where Shipment.status='Delivered(delayed)'");
    mysql_query(connection, rquery);
    sql_result = mysql_store_result(connection);
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%s₩n", sql_row[0]);
    }
    mysql_free_result(sql_result);
    break;
```

->code



```
**Find those packages that were not delivered within the promised time**
Package 12
Package 10
Package 11
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 1', 'International', '11001');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 1', 'Hazardous', '11001');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 3', 'Normal', '11003');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 4', 'International', '11004');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 5', 'Hazardous', '11005');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 5', 'Normal',

'11005');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 7', 'International', '11007');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 8', 'Hazardous', '11008');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 9', 'Normal', '11009');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 10', 'International', '11010');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 11', 'Normal', '11014');

INSERT INTO Package_Info (Package_name, Description, Shipment_ID) VALUES('Package 12', 'Normal', '11002');

-> 배송 상품이 실린 Shipment_ID는 11001,11002,11003,11004,11005,11007,11008,11010,11014이다.

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11001', '20230602', 'In Transit', 50.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11002', '20230601', 'Delivered(delayed)', 75.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11003', '20230603', 'In Transit', 60.25);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11004', '20230604', 'Delivered(on time)', 85.75);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11005', '20230605', 'In Transit', 70.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11007', '20230602', 'In Transit', 80.00);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11008', '20230608', 'Delivered(on time)', 55.50);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11010', '20230410', 'Delivered(delayed)',

95.75);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11014', '20230416', 'Delivered(delayed)', 86.20);

->이와 관련된 Shpment들만 찾아보았을 때 Delivered(delayed)라고 적혀있는 Shipment_ID는 11002, 11010, 11014 이며 이와 연관된 Package name은 Package10, Package11, Package12이다.

따라서 실행결과가 맞다

(Type 5)

Generate the bill for each customer for the past month. Consider creating several types of bills

prj1에서 설명하였던 방법(prj1 보고서 참고)

Simple bill(고객 별 소비 비용)

-Cus_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Shipment set에서 cost를 전부 조회하고 이를 더하여 bill을 청구할 수 있음

service type 별 비용

-Cus_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Ship_Service set에서 각 Shipment ID에 대한 Service_type를 알아내면 Service type별 비용을 청구할 수 있음

각 배송 별 비용

-Cus_ship에서 Customer ID를 조회 후 나오는 Shipment ID를 이용하여, Shipment set에서 각 Shipment ID 별 cost를 bill로 청구할 수 있음

```
with F(FID,NUM) as(
select K.CID ,sum(K.C_COST)
from(select Cus_ship.Customer_ID as CID, Shipment.cost as C_COST
    from Cus_ship natural left outer join Shipment
    where Shipment.date like '20230%')as K
group by K.CID
)

select Customer.name, Customer.address, F.NUM
from Customer,F
where Customer.Customer_ID = F.FID
```

->Simple bill 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table을 이용하여 고객의 ID 기준으로 cost를 합치며 이를 group by를 이용해 묶어주고 F라는 임시 relation을 만든다.(고객의 ID, 고객이 배송한 배송에 대한 비용의 총 합)

Simple bill에서 알고 싶은 내용은 고객의 이름, 고객의 주소, 고객의 주문 비용이므로 Customer과 F를 join하여 두 relation의 ID가 같으면 값을 출력해준다.

*20230 -> 202305 (코드에서는 20230 이후 입력받은 month 데이터를 붙여주어서 20230으로 적어놓았지만, MySQL에서 실행할때는 202305, 202304처럼 특정 month를 붙여주어야함)

```
with F(FID,FTID,FCOST) as(
select K.CID , Ship_Service.service_type_ID,K.C_COST
from(select Cus_ship.Customer_ID as CID, Shipment.Shipment_ID as SID, Shipment.cost as C_COST
    from Cus_ship natural left outer join Shipment
    where Shipment.date like '20230%')as K, Ship_Service
where K.SID= Ship_Service.Shipment_ID
)

select Customer.name, F.FTID, sum(FCOST)
from Customer,F
where Customer.Customer_ID = F.FID
group by Customer.name,F.FTID
```

->A bill listing charges by type of service 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table과 Ship_Service를 이용하여 고객의 배송 번호와 Ship_Service의 배송번호(즉 특정 서비스를 이용하는 배송번호)가 일치하면 F라는 임시 relation을 만든다.(고객 ID, 고객이 주문한 배송이 이용한 서비스 타입 ID, 해당 배송에 대한 COST)

A bill listing charges by type of service에서 알고 싶은 내용은 고객이 배송에서 이용한 서비스 별로 비용을 청구하는 것이므로, Customer와 F를 이용하여 두 개의 ID가 같은 것에 대해 Customer의 이름,서

비스 type ID에 대해 group by를 시켜준다.

*20230 -> 202305

```
with F(FID,FTID,FCOST) as(
select K.CID , K.SID,K.C_COST
from(select Cus_ship.Customer_ID as CID, Shipment.Shipment_ID as SID, Shipment.cost as C_COST
    from Cus_ship natural left outer join Shipment
    where Shipment.date like '20230%')as K

)

select Customer.name, F.FTID, sum(FCOST)
from Customer,F
where Customer.Customer_ID = F.FID
group by Customer.name,F.FTID
```

->An itemize billing listing each individual shipment and the charges for it 구현 내용

with 구문에서 특정 달에 대한 고객의 주문이 담긴 table을 이용하여 F라는 임시 relation을 만든다.(고객 ID, 고객이 주문한 배송 ID, 해당 배송에 대한 COST)

An itemize billing listing each individual shipment and the charges for it에서 알고 싶은 내용은 고객이 배송한 배송번호 별로 비용을 청구하는 것이므로, Customer와 F를 이용하여 두 개의 ID가 같은 것에 대해 Customer의 이름과 Shipment_ID(F.FTID)에 대해 group by를 시켜준다.

*20230 -> 202305

```
case 5:
    printf("---- TYPE 5 ----\n\n");
    printf("**Generate the bill for each customer for the past month. Consider creating several types of bills**\n");
    printf("Which month?(1,2,3,4,5): ");
    scanf("%d", &year);
    sprintf(s1, "%d", year);
    strcpy(rquery, "with F(FID,NUM) as(select K.CID ,sum(K.C_COST)from(select Cus_ship.Customer_ID as CID, Shipment.cost as C_COST from Cus_ship natural left outer join Shipment where Shipment.date like '20230');
    strcat(rquery, s1);
    strcat(rquery, "%')as K group by K.CID order by sum(K.C_COST) desc) select Customer.name, Customer.address, F.NUM from Customer,F  where Customer.Customer_ID = F.FID");
    mysql_query(connection, rquery);
    sql_result = mysql_store_result(connection);
    printf("<Simple bill-Customer name, Address, Amount Owed>\n");
    while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
    {
        printf("%s  %s  %s\n", sql_row[0], sql_row[1], sql_row[2]);
    }
    mysql_free_result(sql_result);
memset(rquery, 0, sizeof(char)*MAXLINE);
strcpy(rquery, "with F(FID,FTID,FCOST) as(select K.CID , Ship_Service.service_type_ID,K.C_COST from(select Cus_ship.Customer_ID as CID, Shipment.Shipment_ID as SID, Shipment.cost as C_COST from Cus_ship natural left outer join Shipment where Shipment.date like '20230');
strcat(rquery, s1);
strcat(rquery, "%')as K, Ship_Service where K.SID= Ship_Service.Shipment_ID) select Customer.name, F.FTID, sum(FCOST) from Customer,F where Customer.Customer_ID = F.FID group by Customer.name,F.FTID");
mysql_query(connection, rquery);
sql_result = mysql_store_result(connection);
printf("<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>\n");
while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
{
    printf("%s  %s  %s\n", sql_row[0], sql_row[1], sql_row[2]);
}
mysql_free_result(sql_result);

memset(rquery, 0, sizeof(char)*MAXLINE);
strcpy(rquery, "with F(FID,FTID,FCOST) as(select K.CID , K.SID,K.C_COST from(select Cus_ship.Customer_ID as CID, Shipment.Shipment_ID as SID, Shipment.cost as C_COST from Cus_ship natural left outer join Shipment where Shipment.date like '20230');
strcat(rquery, s1);
strcat(rquery, "%')as K, Ship_Service where K.SID= Ship_Service.Shipment_ID) select Customer.name, F.FTID, sum(FCOST) from Customer,F where Customer.Customer_ID = F.FID group by Customer.name,F.FTID");
mysql_query(connection, rquery);
sql_result = mysql_store_result(connection);
printf("<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>\n");
while ((sql_row = mysql_fetch_row(sql_result)) != NULL)
{
    printf("%s  %s  %s\n", sql_row[0], sql_row[1], sql_row[2]);
}
mysql_free_result(sql_result);
```

->code

sql query를 만드는 방법은 Type2와 완전히 동일하여 생략

```
**Generate the bill for each customer for the past month. Consider creating several types of bills**
Which month?(1,2,3,4,5): 5
<Simple bill-Customer name, Address, Amount Owed>
John Smith      123 Main Street 22.50
Jane Doe        456 Maple Avenue        12.75
Michael Johnson 789 Oak Road    12.75
<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>
John Smith      20001   10.75
John Smith      20002   11.75
Jane Doe        20003   12.75
Michael Johnson 20003   12.75
<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>
John Smith      11024   10.75
John Smith      11025   11.75
Jane Doe        11026   12.75
Michael Johnson 11027   12.75
Press 0 if you want to select another menu :
```

->실행 결과

실행 결과가 맞는지 dbfiletxt.txt를 보자면 다음과 같다.

<5월의 Shipment ID>

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11024', '20230510', 'Delivered(delayed)', 10.75);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11025', '20230510', 'Delivered(delayed)', 11.75);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11026', '20230510', 'Delivered(delayed)', 12.75);

INSERT INTO Shipment (Shipment_ID, date, status, cost) VALUES('11027', '20230510', 'Delivered(delayed)', 12.75);

<해당 Shipment ID에 대한 Customer ID>

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11024','00001');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11025','00001');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11026','00002');

INSERT INTO Cus_ship(Shipment_ID, Customer_ID) VALUES('11027','00003');

<해당 Customer ID에 대한 Customer name>

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00001', 'John Smith', '123 Main Street', 1111111111, 'Credit Card');

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00002', 'Jane Doe', '456 Maple Avenue', 2222222222, 'Account');

INSERT INTO Customer (Customer_ID, name, address, phone_num, payment_method) VALUES('00003', 'Michael Johnson', '789 Oak Road', 3333333333, 'Cash');

<해당 Shipment ID에 대한 Service_type_ID>

INSERT INTO Ship_Service(Shipment_ID, Service_type_ID) VALUES('11024','20001');

INSERT INTO Ship_Service(Shipment_ID, Service_type_ID) VALUES('11025','20002');

INSERT INTO Ship_Service(Shipment_ID, Service_type_ID) VALUES('11026','20003');

INSERT INTO Ship_Service(Shipment_ID, Service_type_ID) VALUES('11027','20003');

이걸 바탕으로

1. Simple bill

<해당 Shipment ID에 대한 Customer ID>참고

-John Smith: Shipment_ID 11024, 11025이므로 해당 Shipment tuple의 cost 더하면 22.5

-Jane Doe: Shipmnet_ID 11026이므로 해당 Shipment tuple의 cost 더하면 12.75

-Michael Johnson: Shipment_ID 11027이므로 해당 Shipment tuple의 cost 더하면 12.75

*주소는 확인해보면 잘 나오고 있음을 자명하게 알 수 있음

2. A bill listing charges by type of service

<해당 Shipment ID에 대한 Service_type_ID>참고

-John Smith: Shipment_ID 11024, 11025이므로 Service_type_ID는 20001, 20002이며

각 service_type과 연관된 Shipment cost별로 출력이 나오고 있음

-Jane Doe: Shipmnet_ID 11026이므로 Service_type_ID는 20003이며

각 service_type과 연관된 Shipment cost별로 출력이 나오고 있음

-Michael Johnson: Shipment_ID 11027이므로 Service_type_ID는 20003이며

각 service_type과 연관된 Shipment cost별로 출력이 나오고 있음

3. An itemize billing listing each individual shipment and the charges for it

<해당 Shipment ID에 대한 Customer ID> 참고

-John Smith: Shipment_ID 11024, 11025이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

-Jane Doe: Shipmnet_ID 11026이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

-Michael Johnson: Shipment_ID 11027이므로 해당 Shipment tuple 별로 출력값이 나오는 것을 알수 있음

+Type 5에 대한 추가 실행 결과

```
**Generate the bill for each customer for the past month. Consider creating several types of bills**
Which month?(1,2,3,4,5): 4
<Simple bill-Customer name, Address, Amount Owed>
Emily Davis     321 Elm Court   13.75
David Wilson    654 Pine Lane   14.75
Sarah Thompson  987 Cedar Drive 15.75
Christopher Lee 456 Oak Street  16.75
<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>
Emily Davis     20004   13.75
David Wilson    20004   14.75
Sarah Thompson  20005   15.75
Christopher Lee 20006   16.75
<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>
Emily Davis     11028   13.75
David Wilson    11029   14.75
Sarah Thompson  11030   15.75
Christopher Lee 11031   16.75
Press 0 if you want to select another menu :
```

(4월)

```
**Generate the bill for each customer for the past month. Consider creating several types of bills**
Which month?(1,2,3,4,5): 3
<Simple bill-Customer name, Address, Amount Owed>
Olivia Taylor   123 Elm Avenue  136.24
<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>
Olivia Taylor   20007   36.50
Olivia Taylor   20008   49.50
<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>
Olivia Taylor   11013   50.24
Olivia Taylor   11032   17.75
Olivia Taylor   11033   18.75
Olivia Taylor   11034   19.75
Olivia Taylor   11035   29.75
Press 0 if you want to select another menu :
```

(3월)

```
**Generate the bill for each customer for the past month. Consider creating several types of bills**
Which month?(1,2,3,4,5): 2
<Simple bill-Customer name, Address, Amount Owed>
Olivia Taylor   123 Elm Avenue  129.25
Jane Doe        456 Maple Avenue        16.75
<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>
Olivia Taylor   20005   129.25
Jane Doe        20005   16.75
<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>
Olivia Taylor   11036   20.75
Olivia Taylor   11037   14.75
Jane Doe        11038   16.75
Olivia Taylor   11039   93.75
```

(2월)

```
**Generate the bill for each customer for the past month. Consider creating several types of bills**
Which month?(1,2,3,4,5): 1
<Simple bill-Customer name, Address, Amount Owed>
Daniel Anderson 789 Maple Road  117.50
Sophia Martinez 321 Pine Court  82.50
<A bill listing charges by type of service-Customer name, Service_Type_ID, Amount Owed>
Sophia Martinez 20006   42.75
Daniel Anderson 20007   62.75
Sophia Martinez 20009   39.75
<An itemize billing listing each individual shipment and the charges for it-Customer name, Shipment_ID, Amount Owed>
Daniel Anderson 11040   54.75
Sophia Martinez 11041   42.75
Daniel Anderson 11042   62.75
Sophia Martinez 11043   39.75
Press 0 if you want to select another menu :
```

(1월)

(Type 0 and 결과값 출력 후 0입력 구현)

```
case 0:
    return 0;
    break;
}
while (start) {
    printf("Press 0 if you want to select another menu : ");
    scanf("%d", &start);
}
```

->code

Select Query Type에서 0번을 누르면 프로그램이 종료되게 구현하였다. (빨간색 원)

Query 결과가 나오면 0을 누를 때 까지 다시 Select Query Type으로 돌아가지 못하게 while(start){~~}

을 통해 구현하였다.(노란색 원)

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

        1. TYPE 1
        2. TYPE 2
        3. TYPE 3
        4. TYPE 4
        5. TYPE 5
        0. QUIT
Type Select(1,2,3,4,5,0): 0

C:\Users\cw316\source\repos\practice\x64\Debug\practice.exe(20244 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```