REPORT

서강대학교
SOGANG UNIVERSITY

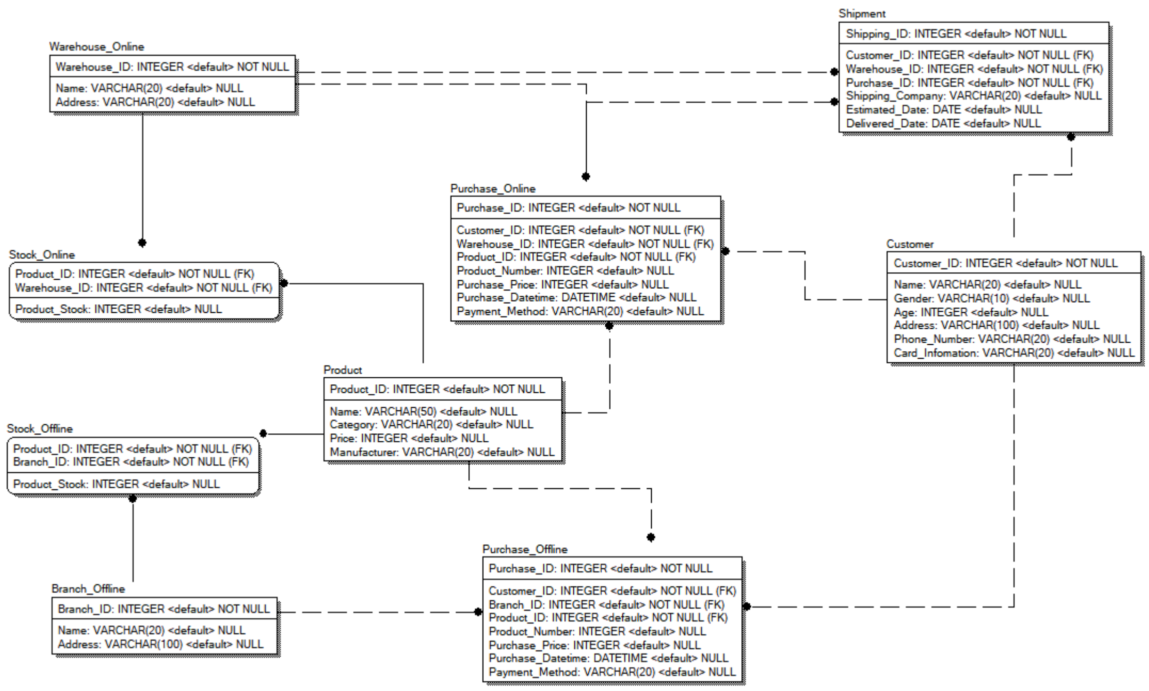
Project 2.
Normalization and Query Processing

과목명	데이터베이스시스템
담당교수	정성원 교수님
학과	컴퓨터공학과
학번	20181702
이름	홍주표
제출일	22.06.07

1. 프로젝트 개요

Project 1 에서 만든 Schema Diagram 의 BCNF 여부를 확인하고 만약 아니라면 BCNF Decomposition 을 수행한다. 또한 명세서에 주어진 Query 13 개를 만족하는 SQL 문을 만들고 C 언어를 통해 만든 데이터에 적용시킨다.

2. BCNF Decomposition


Warehouse_Online
Warehouse_ID: INTEGER <default> NOT NULL
Name: VARCHAR(20) <default> NULL
Address: VARCHAR(20) <default> NULL
Shipment
Shipping_ID: INTEGER <default> NOT NULL
Customer_ID: INTEGER <default> NOT NULL (FK)
Warehouse_ID: INTEGER <default> NOT NULL (FK)
Purchase_ID: INTEGER <default> NOT NULL (FK)
Shipping_Company: VARCHAR(20) <default> NULL
Estimated_Date: DATE <default> NULL
Delivered_Date: DATE <default> NULL
Purchase_Online
Purchase_ID: INTEGER <default> NOT NULL
Customer_ID: INTEGER <default> NOT NULL (FK)
Warehouse_ID: INTEGER <default> NOT NULL (FK)
Product_ID: INTEGER <default> NOT NULL (FK)
Product_Number: INTEGER <default> NULL
Purchase_Price: INTEGER <default> NULL
Purchase_Datetime: DATETIME <default> NULL
Payment_Method: VARCHAR(20) <default> NULL
Stock_Online
Product_ID: INTEGER <default> NOT NULL (FK)
Warehouse_ID: INTEGER <default> NOT NULL (FK)
Product_Stock: INTEGER <default> NULL
Customer
Customer_ID: INTEGER <default> NOT NULL
Name: VARCHAR(20) <default> NULL
Gender: VARCHAR(10) <default> NULL
Age: INTEGER <default> NULL
Address: VARCHAR(100) <default> NULL
Phone_Number: VARCHAR(20) <default> NULL
Card_Infomation: VARCHAR(20) <default> NULL
Product
Product_ID: INTEGER <default> NOT NULL
Name: VARCHAR(50) <default> NULL
Category: VARCHAR(20) <default> NULL
Price: INTEGER <default> NULL
Manufacturer: VARCHAR(20) <default> NULL
Stock_Offline
Product_ID: INTEGER <default> NOT NULL (FK)
Branch_ID: INTEGER <default> NOT NULL (FK)
Product_Stock: INTEGER <default> NULL
Purchase_Offline
Purchase_ID: INTEGER <default> NOT NULL
Customer_ID: INTEGER <default> NOT NULL (FK)
Branch_ID: INTEGER <default> NOT NULL (FK)
Product_ID: INTEGER <default> NOT NULL (FK)
Product_Number: INTEGER <default> NULL
Purchase_Price: INTEGER <default> NULL
Purchase_Datetime: DATETIME <default> NULL
Payment_Method: VARCHAR(20) <default> NULL
Branch_Offline
Branch_ID: INTEGER <default> NOT NULL
Name: VARCHAR(20) <default> NULL
Address: VARCHAR(100) <default> NULL

(1) Warehouse_Online

A. Warehouse_ID -> Name, Address

Warehouse 의 고유번호(ID)가 동일하면 해당 창고의 이름(지점명)과 주소(지명) 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Warehouse_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Warehouse_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

(2) Branch_Offline

A. Branch_ID -> Name, Address

Warehouse_Online 과 동일한 내용이므로 BCNF 를 만족한다.

(3) Product

A. Product_ID -> Name, Cagetory, Price, Manufacturer

Product 의 고유번호(ID)가 동일하면 해당 제품의 이름, 카테고리, 가격, 제조사 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Product_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Product_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

(4) Customer

A. Customer_ID -> Name, Gender, Age, Address, Phone_Number, Card_Information

Customer 의 고유번호(ID)가 동일하면 해당 고객의 이름, 성별, 나이, 주소, 전화번호, 카드정보 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Customer_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Customer_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

(5) Stock_Online

A. Product_ID, Warehouse_ID -> Product_Stock

제품의 고유번호와 Warehouse 의 고유번호가 동일하면 해당 지점의 해당 물품에 대한 재고 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Product_ID, Warehouse_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Product_ID, Warehouse_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

(6) Stock_Offline

A. Product_ID, Branch_ID -> Product_Stock

Stock_Online 과 동일한 내용이므로 BCNF 를 만족한다.

(7) Purchase_Online

A. Purchase_ID -> Customer_ID, Warehouse_ID, Product_ID, Product_Number, Purchase_Price, Purchase_Datetime, Payment_Method

구매이력의 고유번호(ID)가 동일하면 해당 고객의 고유번호, 해당 재고를 관리하는 Warehouse 의 고유번호, 제품의 고유번호, 구매한 제품의 개수, 구매하는 데 지불한 총 비용, 구매한 날짜와 시각, 지불 방법 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Purchase_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Purchase_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

(8) Purchase_Offline

A. Purchase_ID -> Customer_ID, Branch_ID, Product_ID, Product_Number, Purchase_Price, Purchase_Datetime, Payment_Method

Purchase_Online 과 동일한 내용이므로 BCNF 를 만족한다.

(9) Shipment

A. Shipping_ID -> Customer_ID, Warehouse_ID, Purchase_ID, Shipping_Company, Estimated_Date, Delivered_Date

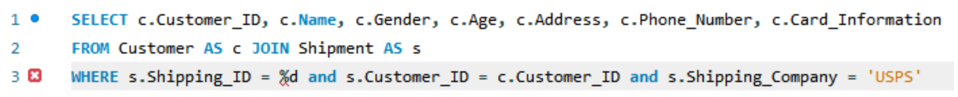
배송의 고유번호(운송장 번호)가 동일하면 고객의 고유번호, Warehouse 의 고유번호, 제품의 고유번호, 배송 회사명, 도착 예정일, 실제 도착일 또한 동일하다. 그러므로 위 Functional Dependency 를 만족한다. Shipping_ID 가 primary key 이므로 super key 이고 BCNF 를 만족한다. Shipping_ID 의 closure 는 릴레이션의 모든 속성을 포함한다.

3. ODBC C Code

C 코드는 모든 query 가 유사한 구성을 가지고 있다.

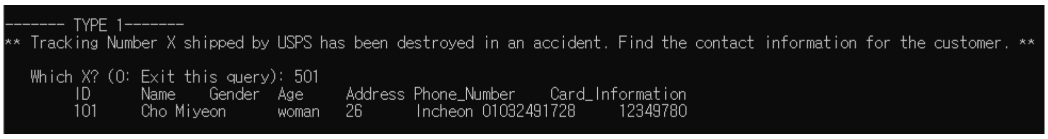
query 라는 MAXLINE(=1000)만큼의 크기를 가지는 일차원 char 배열을 선언하고 sprintf 를 통해 해당 query 에 알맞은 SQL 문을 삽입한다. 그 후 mysql_query, mysql_store_result 함수를 사용하여 결과를 도출하고 mysql_fetch_row 함수를 사용하여 sql_row 변수에 행 별로 값을 할당시킨다. 원하는 기능을 완성시킨 후에는 mysql_free_result 함수를 통해 sql_result 를 free 한다.

1. (TYPE 1) Assume the package shipped by USPS with tracking number X is reported to have been destroyed in an accident. Find the contact information for the customer.



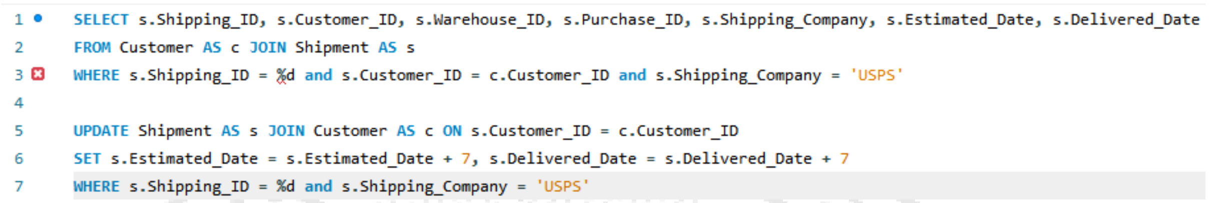
```sql
SELECT c.Customer_ID, c.Name, c.Gender, c.Age, c.Address, c.Phone_Number, c.Card_Information
FROM Customer AS c JOIN Shipment AS s
WHERE s.Shipping_ID = %d and s.Customer_ID = c.Customer_ID and s.Shipping_Company = 'USPS'
```

- %d 에는 입력할 tracking number(X)가 들어간다.

- 

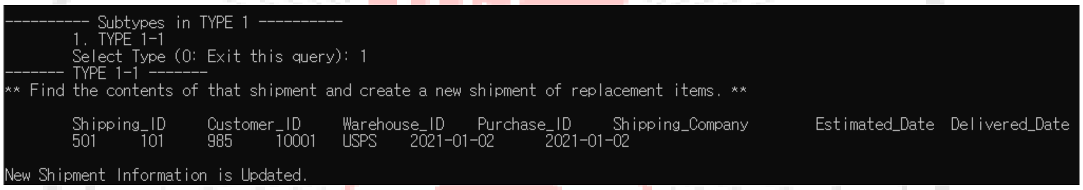
```
------- TYPE 1-------
** Tracking Number X shipped by USPS has been destroyed in an accident. Find the contact information for the customer. **

   Which X? (0: Exit this query): 501
        ID       Name     Gender   Age      Address Phone_Number     Card_Information
        101      Cho Miyeon        woman    26       Incheon 01032491728      12349780
```

2. (TYPE 1-1) Then find the contents of that shipment and create a new shipment of replacement items.



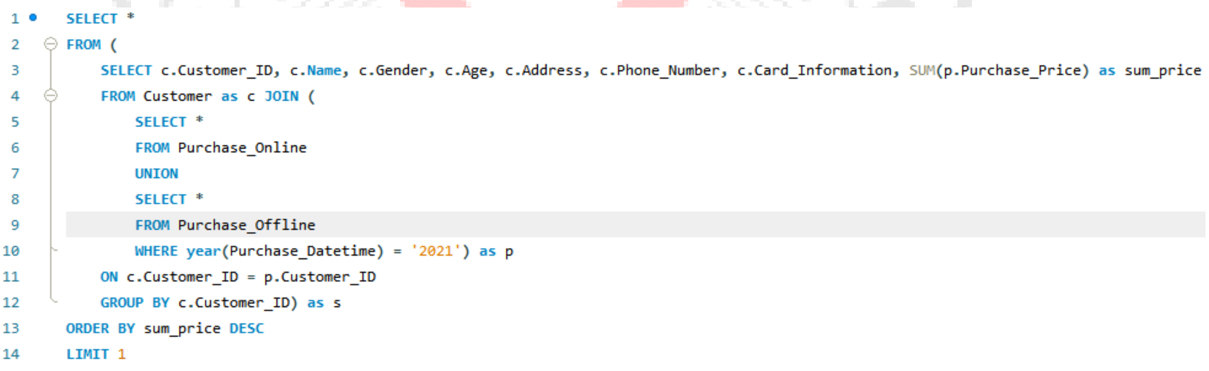
```sql
SELECT s.Shipping_ID, s.Customer_ID, s.Warehouse_ID, s.Purchase_ID, s.Shipping_Company, s.Estimated_Date, s.Delivered_Date
FROM Customer AS c JOIN Shipment AS s
WHERE s.Shipping_ID = %d and s.Customer_ID = c.Customer_ID and s.Shipping_Company = 'USPS'

UPDATE Shipment AS s JOIN Customer AS c ON s.Customer_ID = c.Customer_ID
SET s.Estimated_Date = s.Estimated_Date + 7, s.Delivered_Date = s.Delivered_Date + 7
WHERE s.Shipping_ID = %d and s.Shipping_Company = 'USPS'
```

- 위 세 줄은 content 를 find 하는 SQL 문이고 아래 세 줄은 new shipment 로 replace(update)하는 SQL 문이다.

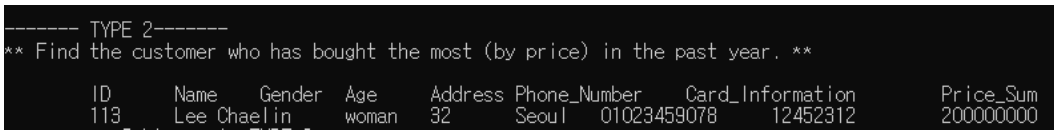
- 

```
---------- Subtypes in TYPE 1 ----------
        1. TYPE 1-1
        Select Type (0: Exit this query): 1
------- TYPE 1-1 -------
** Find the contents of that shipment and create a new shipment of replacement items. **

        Shipping_ID     Customer_ID     Warehouse_ID     Purchase_ID     Shipping_Company        Estimated_Date  Delivered_Date
        501     101     985     10001   USPS    2021-01-02      2021-01-02

New Shipment Information is Updated.
```

3. (TYPE 2)Find the customer who has bought the most (by price) in the past year.

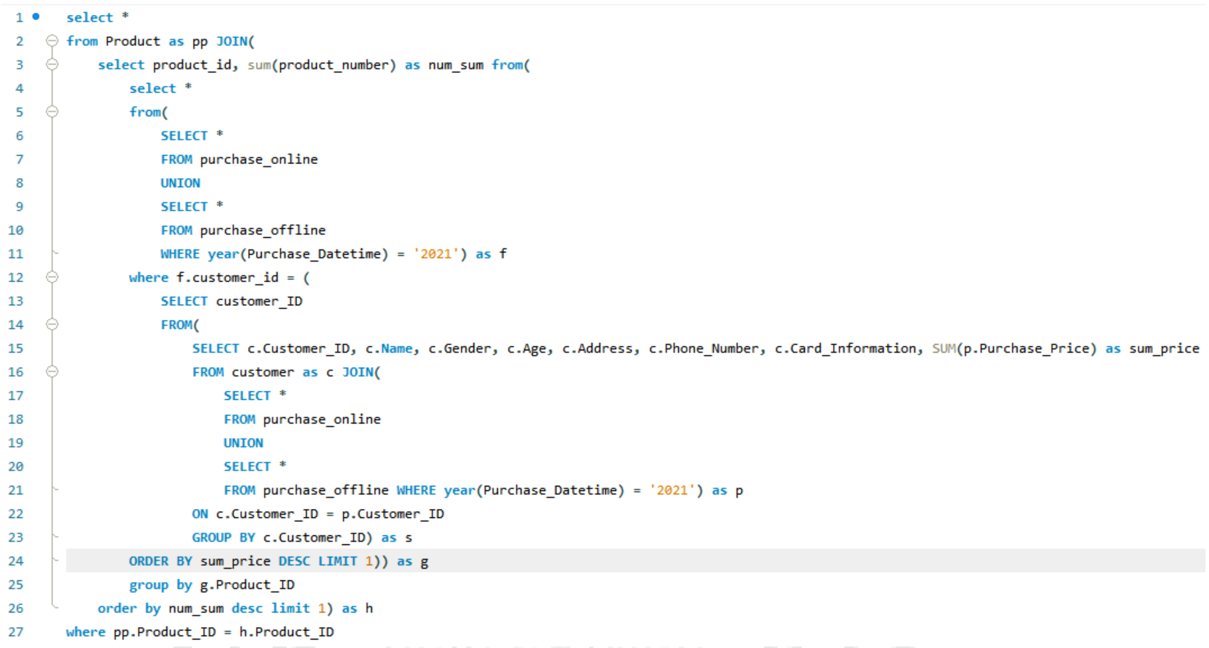


```sql
SELECT *
FROM (
    SELECT c.Customer_ID, c.Name, c.Gender, c.Age, c.Address, c.Phone_Number, c.Card_Information, SUM(p.Purchase_Price) as sum_price
    FROM Customer as c JOIN (
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline
        WHERE year(Purchase_Datetime) = '2021') as p
    ON c.Customer_ID = p.Customer_ID
    GROUP BY c.Customer_ID) as s
ORDER BY sum_price DESC
LIMIT 1
```

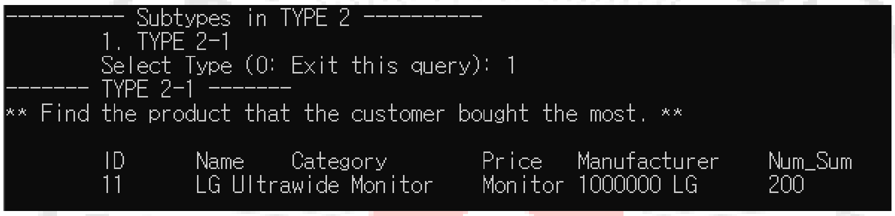
- 

```
------- TYPE 2-------
** Find the customer who has bought the most (by price) in the past year. **

        ID      Name     Gender   Age      Address Phone_Number     Card_Information        Price_Sum
        113     Lee Chaelin       woman    32       Seoul    01023459078      12452312        200000000
```

4. (TYPE 2-1)Then find the product that the customer bought the most.



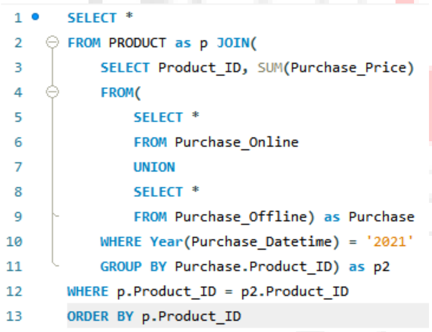
```sql
select *
from Product as pp JOIN(
    select product_id, sum(product_number) as num_sum from(
        select *
        from(
            SELECT *
            FROM purchase_online
            UNION
            SELECT *
            FROM purchase_offline
            WHERE year(Purchase_Datetime) = '2021') as f
        where f.customer_id = (
            SELECT customer_ID
            FROM(
                SELECT c.Customer_ID, c.Name, c.Gender, c.Age, c.Address, c.Phone_Number, c.Card_Information, SUM(p.Purchase_Price) as sum_price
                FROM customer as c JOIN(
                    SELECT *
                    FROM purchase_online
                    UNION
                    SELECT *
                    FROM purchase_offline WHERE year(Purchase_Datetime) = '2021') as p
                ON c.Customer_ID = p.Customer_ID
                GROUP BY c.Customer_ID) as s
        ORDER BY sum_price DESC LIMIT 1)) as g
        group by g.Product_ID
    order by num_sum desc limit 1) as h
where pp.Product_ID = h.Product_ID
```



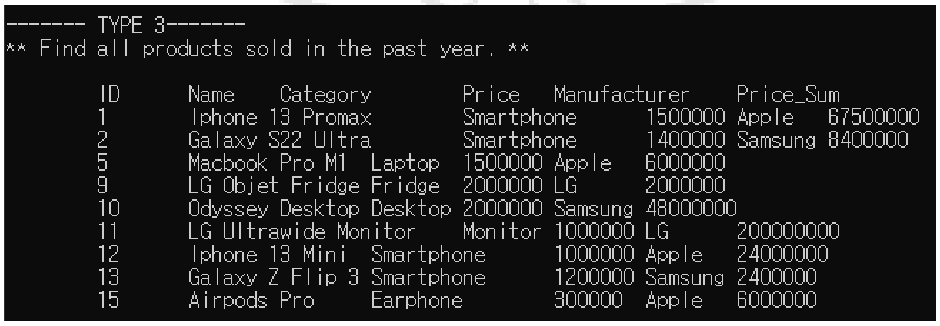
```
---------- Subtypes in TYPE 2 ----------
        1. TYPE 2-1
        Select Type (0: Exit this query): 1
------- TYPE 2-1 -------
** Find the product that the customer bought the most. **

        ID      Name     Category         Price    Manufacturer     Num_Sum
        11      LG Ultrawide Monitor      Monitor  1000000 LG       200
```

-

5. (TYPE 3) Find all products sold in the past year.



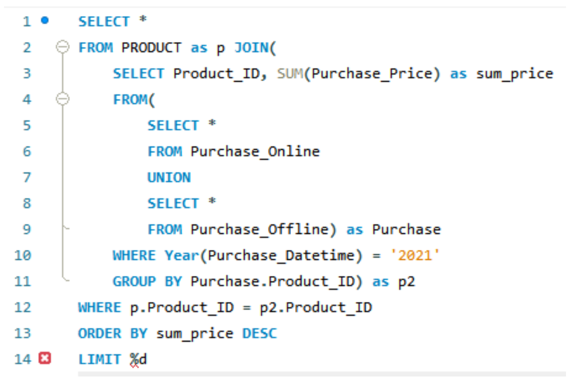
```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Purchase_Price)
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID
ORDER BY p.Product_ID
```



```
------- TYPE 3-------
** Find all products sold in the past year. **

        ID      Name     Category         Price    Manufacturer     Price_Sum
        1       Iphone 13 Promax          Smartphone       1500000 Apple    67500000
        2       Galaxy S22 Ultra          Smartphone       1400000 Samsung 8400000
        5       Macbook Pro M1   Laptop   1500000 Apple    6000000
        9       LG Objet Fridge Fridge    2000000 LG       2000000
        10      Odyssey Desktop Desktop 2000000 Samsung 48000000
        11      LG Ultrawide Monitor      Monitor 1000000 LG       200000000
        12      Iphone 13 Mini   Smartphone       1000000 Apple    24000000
        13      Galaxy Z Flip 3 Smartphone        1200000 Samsung 2400000
        15      Airpods Pro      Earphone         300000  Apple    6000000
```

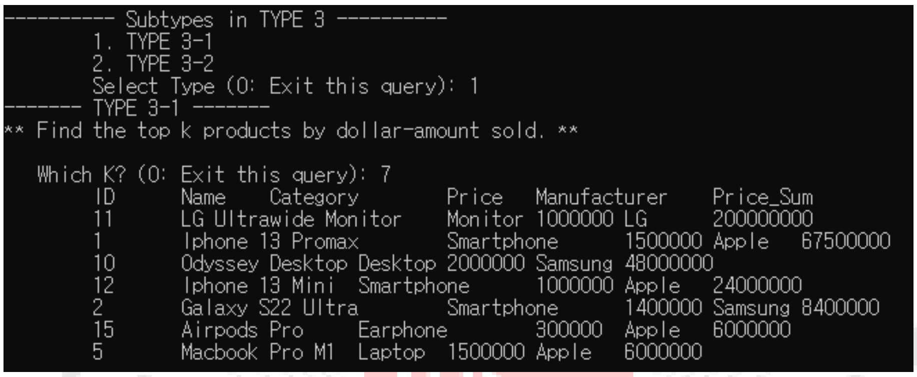
-

6. (TYPE 3-1)Then find the top k products by dollar-amount sold.



```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Purchase_Price) as sum_price
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID
ORDER BY sum_price DESC
LIMIT %d
```

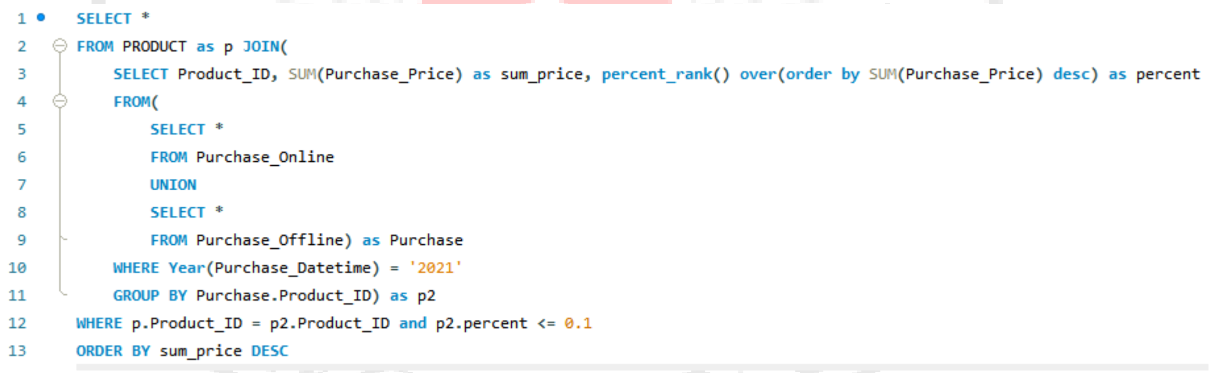
- %d 에는 입력할 Top k Products(k)가 들어간다.



```
---------- Subtypes in TYPE 3 ----------
        1. TYPE 3-1
        2. TYPE 3-2
        Select Type (0: Exit this query): 1
------- TYPE 3-1 -------
** Find the top k products by dollar-amount sold. **

   Which K? (0: Exit this query): 7
        ID      Name    Category        Price   Manufacturer    Price_Sum
        11      LG Ultrawide Monitor    Monitor 1000000 LG      200000000
        1       Iphone 13 Promax        Smartphone      1500000 Apple   67500000
        10      Odyssey Desktop Desktop 2000000 Samsung 48000000
        12      Iphone 13 Mini  Smartphone      1000000 Apple   24000000
        2       Galaxy S22 Ultra        Smartphone      1400000 Samsung 8400000
        15      Airpods Pro     Earphone        300000  Apple   6000000
        5       Macbook Pro M1  Laptop  1500000 Apple   6000000
```

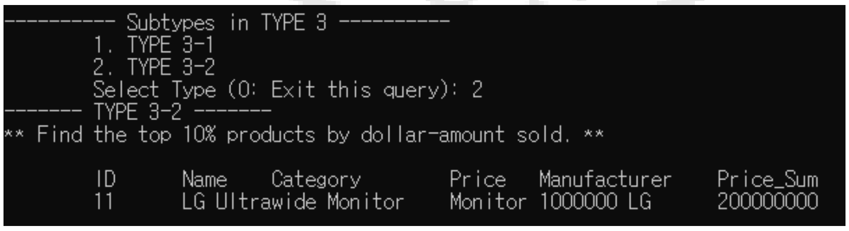
-

7. (TYPE 3-2)And then find the top 10% products by dollar-amount sold.



```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Purchase_Price) as sum_price, percent_rank() over(order by SUM(Purchase_Price) desc) as percent
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID and p2.percent <= 0.1
ORDER BY sum_price DESC
```

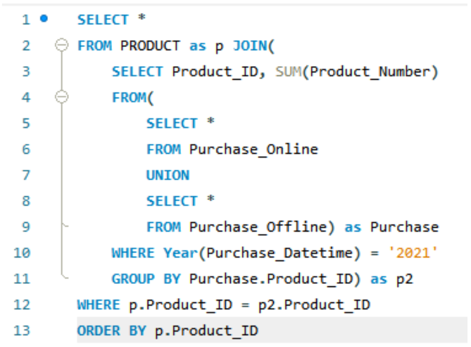
- Percent_rank()를 사용하여 상위 10%에 해당하는 자료를 도출했다.



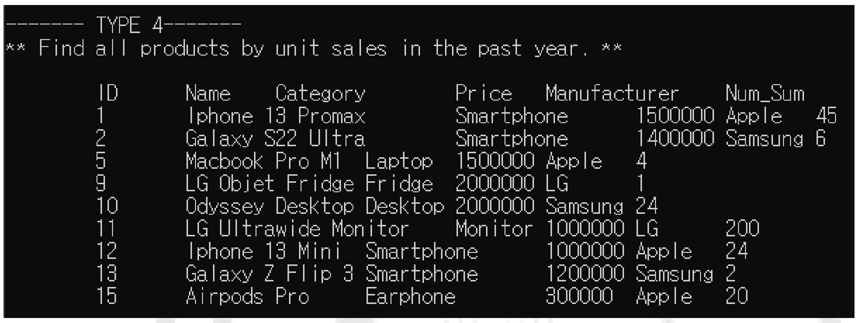
```
---------- Subtypes in TYPE 3 ----------
        1. TYPE 3-1
        2. TYPE 3-2
        Select Type (0: Exit this query): 2
------- TYPE 3-2 -------
** Find the top 10% products by dollar-amount sold. **

        ID      Name    Category        Price   Manufacturer    Price_Sum
        11      LG Ultrawide Monitor    Monitor 1000000 LG      200000000
```

-

8. (TYPE 4) Find all products by unit sales in the past year.

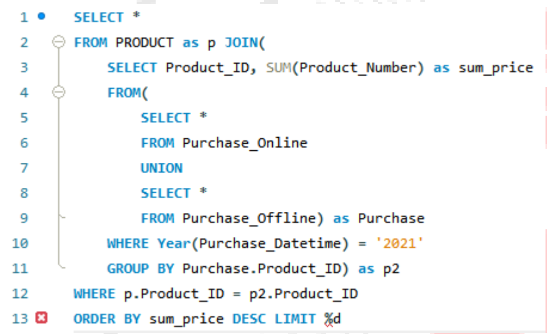


```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Product_Number)
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID
ORDER BY p.Product_ID
```

- 

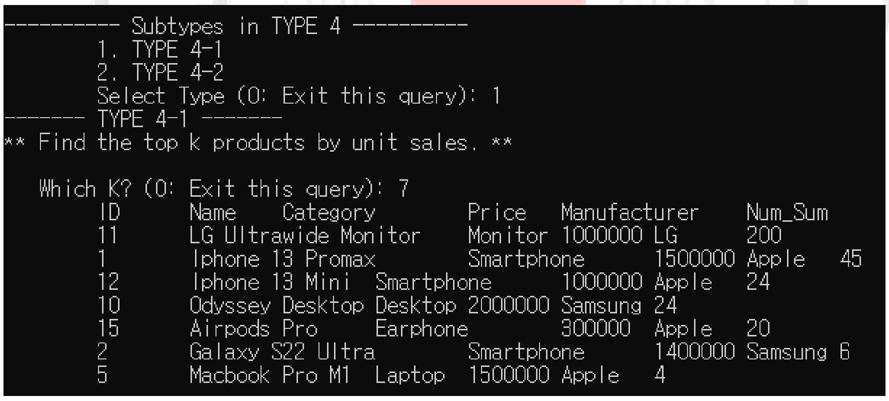
```
------- TYPE 4-------
** Find all products by unit sales in the past year. **

        ID      Name    Category        Price   Manufacturer    Num_Sum
        1       Iphone 13 Promax        Smartphone      1500000 Apple   45
        2       Galaxy S22 Ultra        Smartphone      1400000 Samsung 6
        5       Macbook Pro M1  Laptop  1500000 Apple   4
        9       LG Objet Fridge Fridge  2000000 LG      1
        10      Odyssey Desktop Desktop 2000000 Samsung 24
        11      LG Ultrawide Monitor    Monitor 1000000 LG      200
        12      Iphone 13 Mini  Smartphone      1000000 Apple   24
        13      Galaxy Z Flip 3 Smartphone      1200000 Samsung 2
        15      Airpods Pro     Earphone        300000  Apple   20
```

9. (TYPE 4-1) Then find the top k products by unit sales.



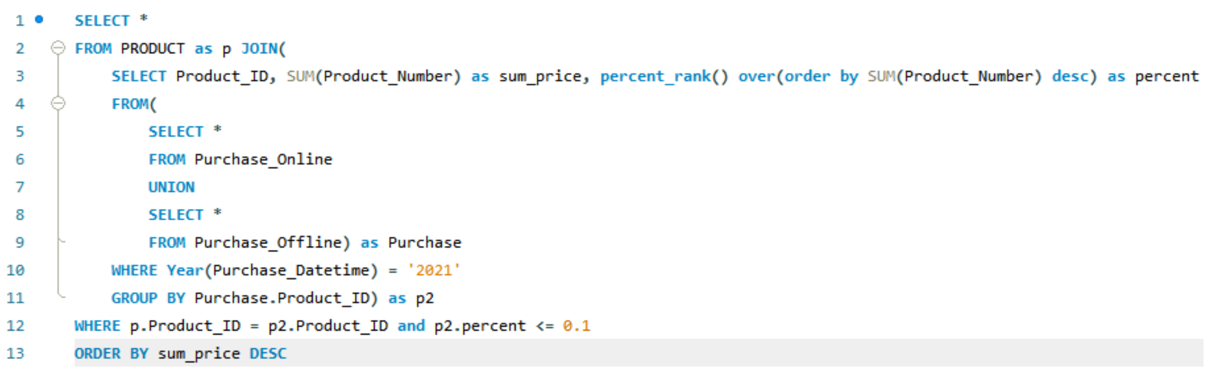
```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Product_Number) as sum_price
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID
ORDER BY sum_price DESC LIMIT %d
```

- %d 에는 입력할 Top k Products(k)가 들어간다.

- 

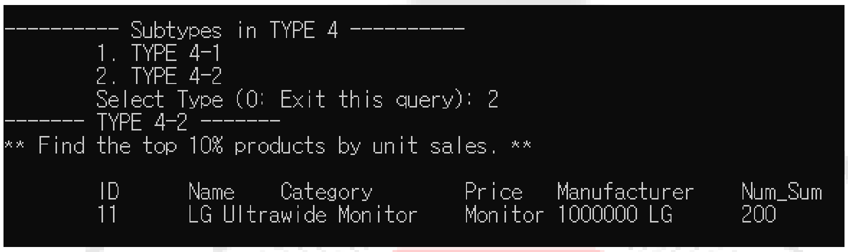
```
---------- Subtypes in TYPE 4 ----------
        1. TYPE 4-1
        2. TYPE 4-2
        Select Type (0: Exit this query): 1
------- TYPE 4-1 -------
** Find the top k products by unit sales. **

   Which K? (0: Exit this query): 7
        ID      Name    Category        Price   Manufacturer    Num_Sum
        11      LG Ultrawide Monitor    Monitor 1000000 LG      200
        1       Iphone 13 Promax        Smartphone      1500000 Apple   45
        12      Iphone 13 Mini  Smartphone      1000000 Apple   24
        10      Odyssey Desktop Desktop 2000000 Samsung 24
        15      Airpods Pro     Earphone        300000  Apple   20
        2       Galaxy S22 Ultra        Smartphone      1400000 Samsung 6
        5       Macbook Pro M1  Laptop  1500000 Apple   4
```

10. (TYPE 4-2) And then find the top 10%products by unit sales.



```sql
SELECT *
FROM PRODUCT as p JOIN(
    SELECT Product_ID, SUM(Product_Number) as sum_price, percent_rank() over(order by SUM(Product_Number) desc) as percent
    FROM(
        SELECT *
        FROM Purchase_Online
        UNION
        SELECT *
        FROM Purchase_Offline) as Purchase
    WHERE Year(Purchase_Datetime) = '2021'
    GROUP BY Purchase.Product_ID) as p2
WHERE p.Product_ID = p2.Product_ID and p2.percent <= 0.1
ORDER BY sum_price DESC
```

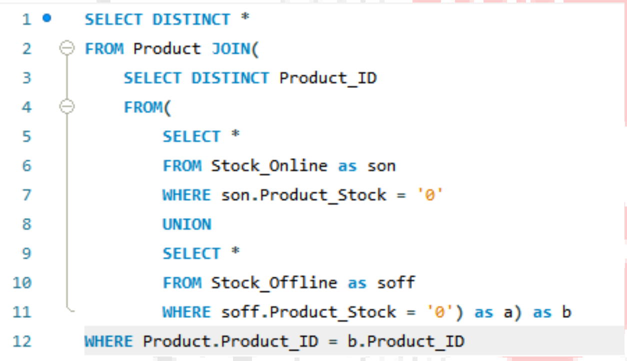
- Percent_rank()를 사용하여 상위 10%에 해당하는 자료를 도출했다.



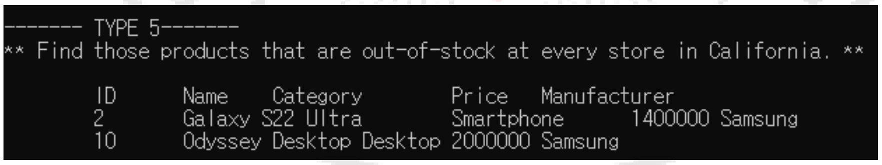
```
---------- Subtypes in TYPE 4 ----------
        1. TYPE 4-1
        2. TYPE 4-2
        Select Type (0: Exit this query): 2
------- TYPE 4-2 -------
** Find the top 10% products by unit sales. **

        ID      Name    Category        Price   Manufacturer    Num_Sum
        11      LG Ultrawide Monitor    Monitor 1000000 LG      200
```

-

11. (TYPE 5) Find those products that are out-of-stock at every store in California.



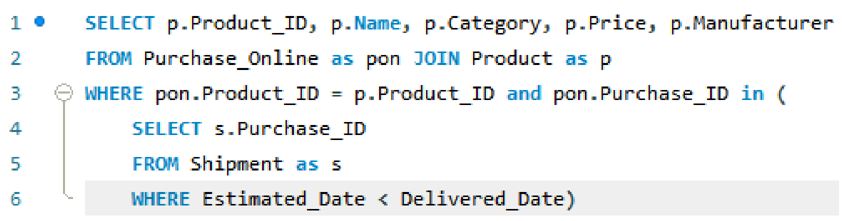
```sql
SELECT DISTINCT *
FROM Product JOIN(
    SELECT DISTINCT Product_ID
    FROM(
        SELECT *
        FROM Stock_Online as son
        WHERE son.Product_Stock = '0'
        UNION
        SELECT *
        FROM Stock_Offline as soff
        WHERE soff.Product_Stock = '0') as a) as b
WHERE Product.Product_ID = b.Product_ID
```



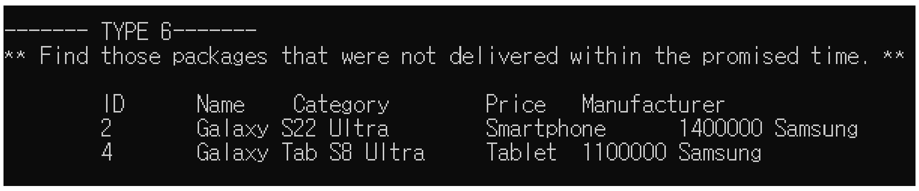
```
------- TYPE 5-------
** Find those products that are out-of-stock at every store in California. **

        ID      Name    Category        Price   Manufacturer
        2       Galaxy S22 Ultra        Smartphone      1400000 Samsung
        10      Odyssey Desktop Desktop 2000000 Samsung
```

-

12. (TYPE 6) Find those packages that were not delivered within the promised time.



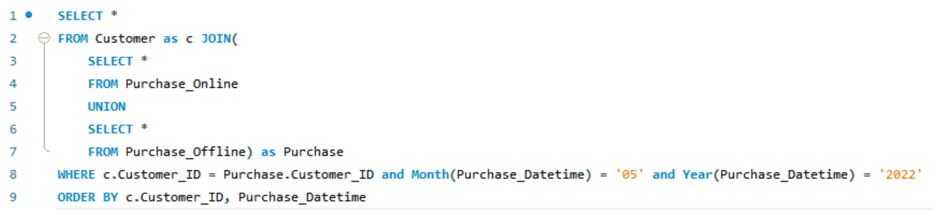
```sql
SELECT p.Product_ID, p.Name, p.Category, p.Price, p.Manufacturer
FROM Purchase_Online as pon JOIN Product as p
WHERE pon.Product_ID = p.Product_ID and pon.Purchase_ID in (
    SELECT s.Purchase_ID
    FROM Shipment as s
    WHERE Estimated_Date < Delivered_Date)
```



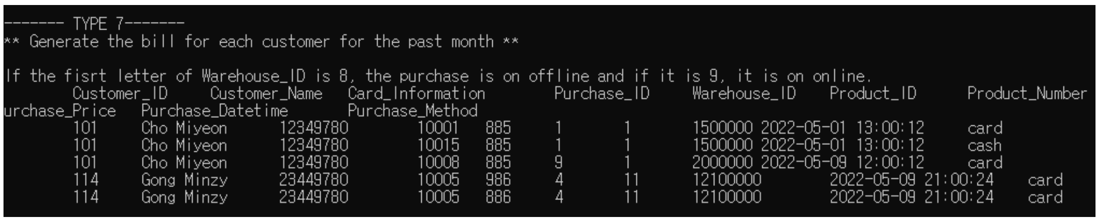
```
------- TYPE 6-------
** Find those packages that were not delivered within the promised time. **

        ID      Name    Category        Price   Manufacturer
        2       Galaxy S22 Ultra        Smartphone      1400000 Samsung
        4       Galaxy Tab S8 Ultra     Tablet  1100000 Samsung
```

-

13. (TYPE 7) Generate the bill for each customer for the past month.



```sql
SELECT *
FROM Customer as c JOIN(
    SELECT *
    FROM Purchase_Online
    UNION
    SELECT *
    FROM Purchase_Offline) as Purchase
WHERE c.Customer_ID = Purchase.Customer_ID and Month(Purchase_Datetime) = '05' and Year(Purchase_Datetime) = '2022'
ORDER BY c.Customer_ID, Purchase_Datetime
```



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

If the fisrt letter of Warehouse_ID is 8, the purchase is on offline and if it is 9, it is on online.
        Customer_ID     Customer_Name   Card_Information        Purchase_ID     Warehouse_ID    Product_ID      Product_Number
urchase_Price   Purchase_Datetime       Purchase_Method
        101     Cho Miyeon      12349780        10001   885     1       1       1500000 2022-05-01 13:00:12     card
        101     Cho Miyeon      12349780        10015   885     1       1       1500000 2022-05-01 13:00:12     cash
        101     Cho Miyeon      12349780        10008   885     9       1       2000000 2022-05-09 12:00:12     card
        114     Gong Minzy      23449780        10005   986     4       11      12100000        2022-05-09 21:00:24     card
        114     Gong Minzy      23449780        10005   886     4       11      12100000        2022-05-09 21:00:24     card
```

-