

Project 2. Normalization and Query Processing

Electronics Vendor company

목차

1. Physical Schema Diagram

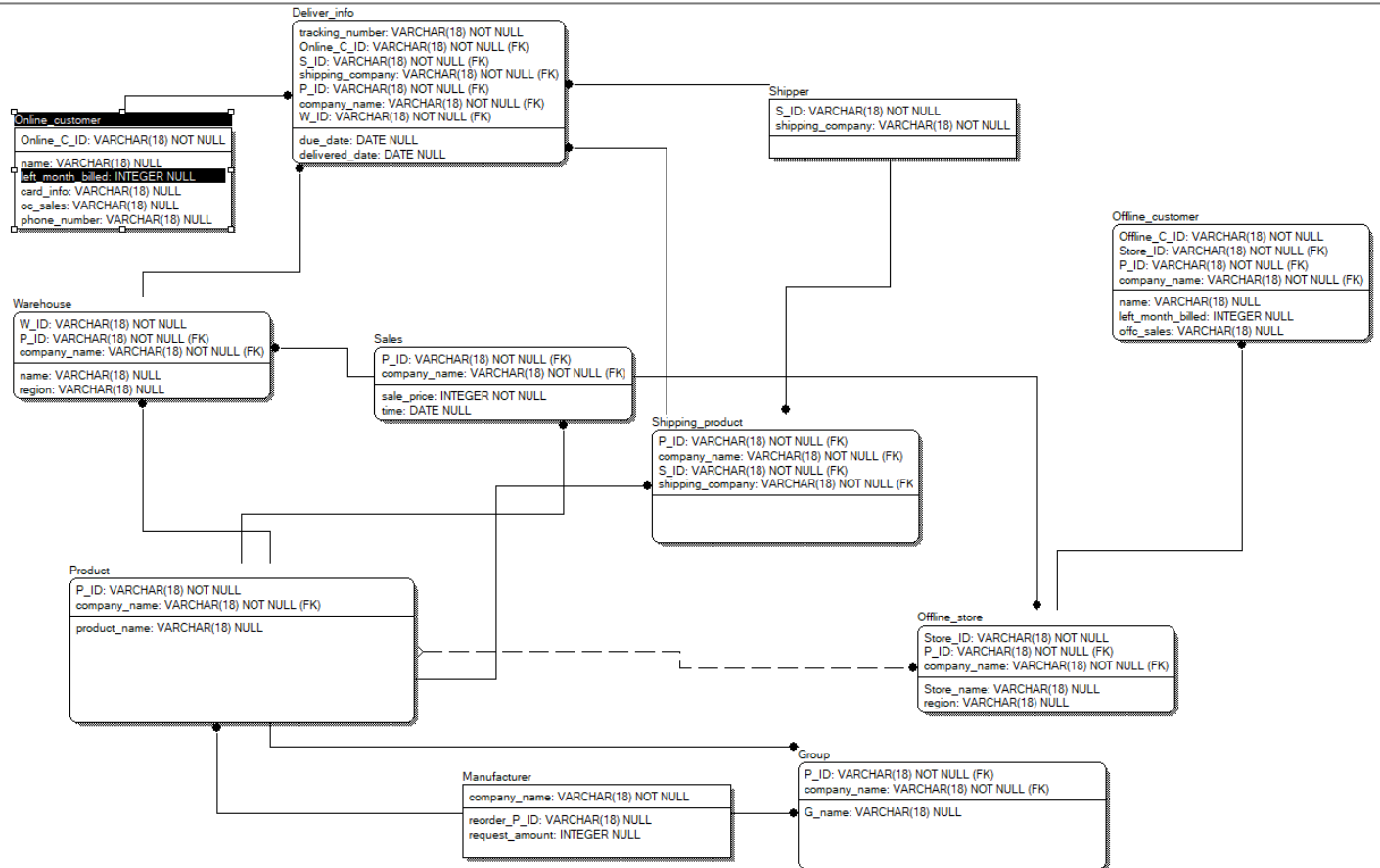
2. BCNF Decomposition

3. ODBC C code

3-1. description

3-2. query result example

1. Physical Schema Diagram



Entities

Manufacturer

reorder을 위한 entity이다. 조건에 따르면 제조사는 reorder가 들어오면 이를 확인하여 공급해야 하므로 reorder product와 request를 담는 reorder_P_ID와 reorder_request를 포함하였다.

Group

product를 reference하며 G_ID가 PK이다. G_name이 존재하는데 예를 들면 데스크탑, 모니터, 마우스, 키보드를 합쳐 컴퓨터 세트라는 이름으로 명명할 수 있다.

Warehouse

W_ID를 primary key로 가지고, product를 reference한다. ware하우스의 이름과 위치를 저장할 수 있도록 name,region을 추가했다.

Product

Manufacture를 reference하여 제조사는 FK이다. product ID는 고유값으로 하기 위해 primary key로 지정했다. 반면 product_name은 primary key가 아닌데, 애플에서 출시한 아이폰을 예로 들면, 기기의 시리얼 넘버가 P_ID가 될 것이고, 아이폰이 product_name이 된다.

Sales

P_ID, company_name을 FK로 가지고 고유의 값은 sale_price와 time이다. 판매 정보를 저장하는 entity인데, 상품이 판매되면 판매된 시간과 판매 가격, 어떤 상품이 판매되었는지 '판매' 기록을 저장하기 위해 생성했다.

Offline_store

Store_ID를 primary key로 가지고 P_ID, company_name은 product entity에서 참조한 FK이다. query문의 california 조건을 위해 region 정보를 저장하도록 만들었다.

Offline_customer

Customer ID와 name, 잔여 할부개월 수, 소비한 판매 금액을 저장한다. 이전의 project와 달라진 점은, 기존에 store에 고객 정보를 저장하는 방식에서 Offline_customer entity에 store 정보를 저장하는 식으로 바꿨다.

Online_customer

Customer ID, name, 남은 결제 개월 수, 카드 정보와 함께 현재까지 각 고객이 소비한 판매 금액을 oc_sales에 저장한다. Query의 조건을 위해 phone number를 저장한다. 이는 상품이 배송되지 않을 경우 사용된다. 외래키를 하나도 가지고 있지 않는 entity이다.

Shipper

구분을 위해 Shipper ID와 shipping company를 primary key로 가진다.

Shipping_product

Product entity에 shipping 정보를 외래키로 가지지 않기 위해 만든 entity이다. product entity와 shipper entity를 참조한다.

Deliver_info

기존의 Deliver entity의 의미를 명확히 하기 위해 이름을 변경했다. query의 조건인, 배송이 늦게 오거나 배송 되지 않은 항목을 찾기 위해 배송 예정일과 실제 배송일을 추가했다. 배송이 오지 않은 경우 실제 배송일이 NULL 이 된다.

Relations

기존의 relation에서 bcnf를 만족하기 위해, query와 데이터에 알맞게 변경된 relation이 존재한다.

Product to Warehouse

새로 추가된 relation이다. 재고 관리를 위해 Warehouse에 상품 정보가 존재해야 하기 때문에 추가했다.

Product to Group

기존의 product_making relation을 수정했다. Group에 여러 가지 product가 존재할 수 있도록 relation을 만들었다.

Product to Shipping_product

Deliver_info의 BCNF를 위해 생성한 관계이다.

Product to Offline_store

Offline store에 있는 product의 재고를 관리할 수 있도록 생성한 relation이다.

Product to Sales

새로 추가된 relation이다. 기존에는 product에 sales 정보를 저장했는데, BCNF를 위해 이 둘을 분리해서 sales 정보와 product 정보를 각각 따로 나눴다.

Online_customer to Deliver_info

기존의 deliver_customer relation과 같다.

Offline_store to Offline_customer

Offline store가 저장하는 offline customer 들이다. Store은 구매자의 정보를 저장하는 것이다.

Shipper to Deliver_info

기존의 Ship_deliver 관계와 같다.

Shipper to Shipping_product

BCNF를 위해 새로 추가된 relation이다.

Manufacturer to Product

BCNF를 위해 product에 저장하던 제조사 정보를 manufacturer로 분리했다.

Manufacturer to Group

Group과 product 간 관계에서 생기는 BCNF 위반을 없애기 위해 생성한 관계이다.

Shipping_product to Deliver_info

Deliver info의 BCNF 위반을 없애기 위해 생성한 관계이다.

Sales to Offline_store

Offline store의 sales 정보를 위한 관계이다.

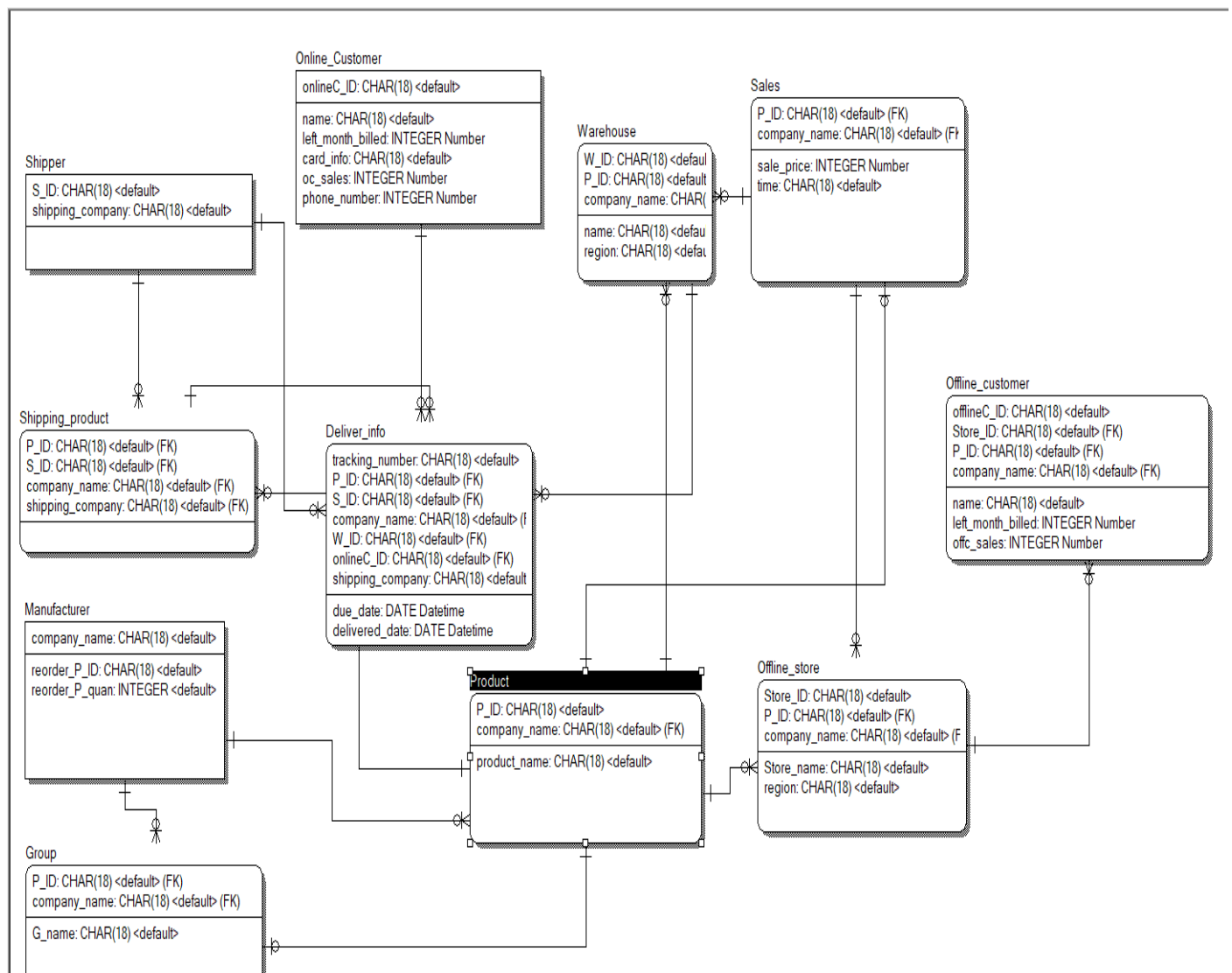
Sales to Warehouse

기존의 warehouse에 sales 정보를 추가하기 위해 생성한 관계이다.

Warehouse to Deliver_info

warehouse에서 전달한, 즉 소모한 재고를 관리하기 위해 생성한 관계이다.

2.BCNF Decomposition



BCNF를 만들기 위해서 entity들을 수정했다. 먼저 분리될 수 있는 Product와 sales를 분리하여 sales에 product의 가격을 포함하도록 했다. Product는 상품 그 자체만 포함하고, 가격은 sales에서 담당하는 식으로 만들었다.

각각의 entity별 functional dependency를 통한 BCNF 확인은 다음과 같다.

1. Shipper: S_ID, shipping_company -> S_ID, shipping_company

S_ID와 shipping company는 각각 동일하다. 결정자와 종속자가 같으므로 BCNF를 만족한다.

2. Online_customer: C_ID -> name, left_month_billed, card_info, oc_sales, phone_number

소비자의 id가 같으면 이름, 잔여할부개월, 카드정보, 판매금액, 전화번호가 같다. 함수적 종속성이 존재한다.

3. Warehouse: W_ID, P_ID, company_name -> name, region

창고의 w_id와 창고가 보관하는 상품의 id, 회사 이름이 같으면 그것은 같은 지역의 같은 이름을 가지는 창고이다. (= 동일한 창고이다.) 함수적 종속성이 존재한다.

4. Sales: P_ID -> company_name

Product ID는 같은 company name이라 할지라도 모두 다르기 때문에 각각의 상품이 팔린 가격과 시간은 모두 다르다. 즉 a -> b이면 b가 a에 포함되는 식의 함수적 종속성이 존재한다.

5. Shipping_product: P_ID, S_ID, company_name, shipping_company -> P_ID, S_ID, company_name, shipping_company

결정자와 종속자가 같으므로 BCNF를 만족한다.

6. Deliver_info: tracking_number, P_ID, S_ID, company_name, W_ID, C_ID, shipping_company -> due_date, delivered_date

배송 정보가 같으면 동일한 도착시간/도착예정시간을 표시해야 한다. 함수적 종속성이 존재한다.

7. Offline_customer: C_ID, Store_ID, P_ID, company_name -> name, left_month_billed, offc_sales

고객의 id와 store id, 구입한 상품의 이름과 회사가 같다면 고객의 이름과 잔여할부개월수, 누적결제금액이 동일하다. 고객이 만일 다른 가게에서 구입하고 store_id가 다를 수 있는 등 왼쪽의 항의 가짓수가 오른쪽의 항의 가짓수보다 더 많으므로 a->b에서 b가 a에 포함되는 함수적 종속성이 존재한다.

8. Manufacturer: company_name -> reorder_P_ID, reorder_P_quan

회사별로 reorder의 종류가 여러가지 존재할 수 있기 때문에 a->b에서 b가 a에 포함되는 함수적 종속성이 존재한다.

9. Product: P_ID, company_name -> product_name

여기서 product id는 제품마다 부여되는 시리얼 넘버 개념으로, 제품마다 고유의 값을 가지는 pkey이다. 이를 바탕으로, product name이 같아도 수많은 다른 제품(ex 아이폰)이 존재하기 때문에 a->b에서 b가 a에 포함되는 함수적 종속성이 존재한다.

10. Offline_store: Store_ID, P_ID, company_name -> Store_name, region

Store에 여러가지 상품이 존재할 수 있기 때문에 a->b 관계에서 a가 b보다 더 많은 값을 가지며 b가 a

에 포함되는 함수적 종속성이 존재한다.

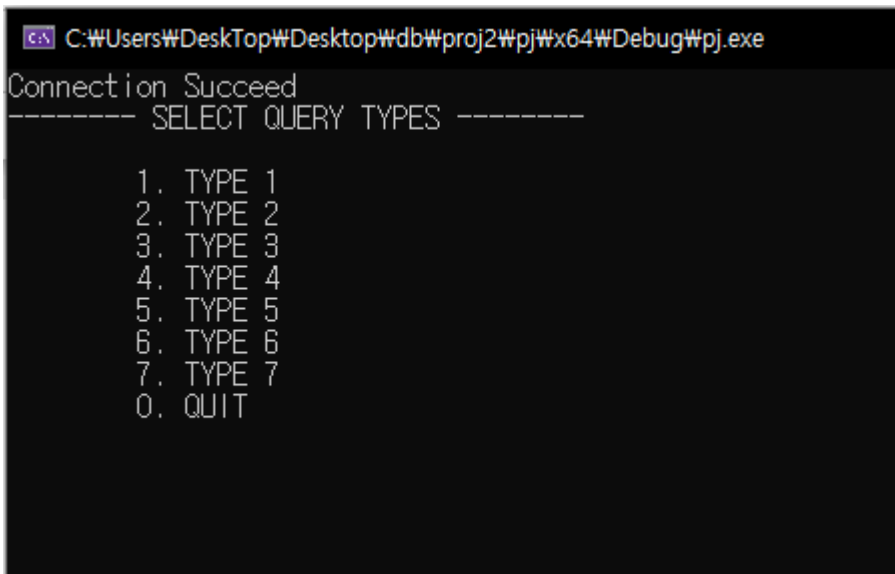
11. Group: P_ID, company_name -> G_name

논리구조상, 예를 들면 컴퓨터 세트를 생각했을 때, 키보드와 모니터, 마우스는 각각 다른 product id를 가질 수 있기 때문에 a->b에서 b가 a에 종속되는 함수적 종속성을 가진다.

3.ODBC C code

완성한 항목을 첨부하면 3번과 4번을 완성했다. 나머지는 Mysql workbench에 제대로 table이 업로드가 되지 않는 부분을 수정하지 못해서 업로드가 되는 부분으로 완성할 수 있는 항목을 만들었다.

입력 화면은 다음과 같다.



```
C:\Users\DeskTop\Desktop\db\proj2\pj\#x64\Debug\pj.exe
Connection Succeed
----- SELECT QUERY TYPES -----
1. TYPE 1
2. TYPE 2
3. TYPE 3
4. TYPE 4
5. TYPE 5
6. TYPE 6
7. TYPE 7
0. QUIT
```

Query 부분만을 살펴보면 (sprint 부분)

Find all products sold in the past year.

```
SELECT * from sales where YEAR(time) = '2021'
```

올해가 2022년이므로 sales table의 time이 2021인 항목들을 모두 선택한다.

Then find the top k products by dollar-amount sold.

3-1. "SELECT * from sales where YEAR(time) = '2021' order by sale_price desc limit %d",
k

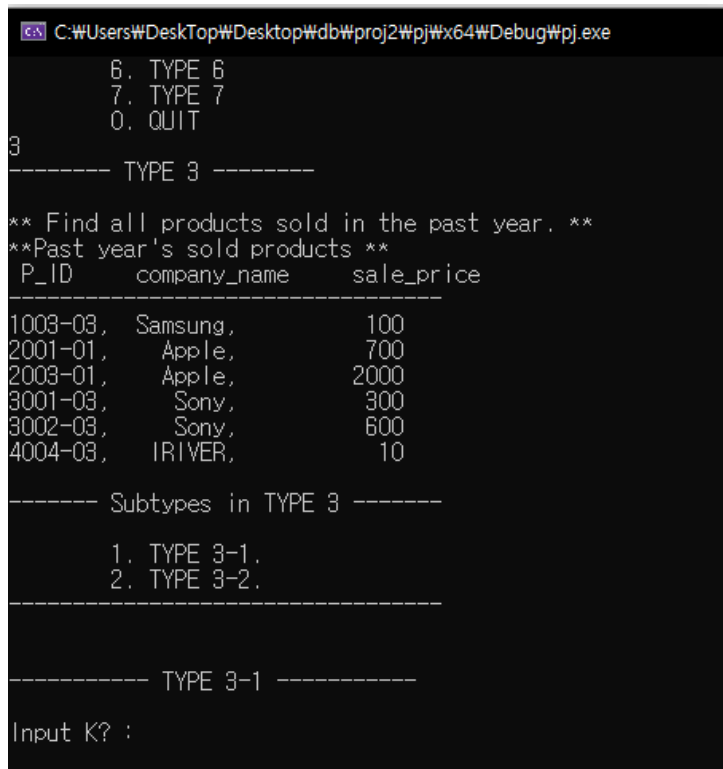
If k == 30이면 order by sale_price desc limit 30이 입력되는 것과 같이 상위 k개를 정렬할 수 있도록 쿼리를 만들었다.

And then find the top 10% products by dollar-amount sold.

```
3-2. select * from (select P_ID, company_name, sale_price, PERCENT_RANK() over(order by
sale_price desc) as per_rank from sales where YEAR(time) = '2021') a where a.per_rank
<= 0.1
```

여기서 불필요한 per_rank가 포함되는데 이것은 출력할 때, 마지막 sql_row 항목을 포함하지 않음으로써 결과 화면에 출력하지 않는다. 먼저 3-1처럼 내림차순으로 sale_price를 정렬한 다음 per_rank에 각각의 누적을 저장 (0 ~1) 한 뒤 이것이 0.1 이하인 것, 즉 상위 10%에 해당하는 column들을 추출한다.

입력 창의 다음과 같다.



```
C:\Users\Desktop\Desktop\pdb\proj2\pj\#x64\Debug\pj.exe
6. TYPE 6
7. TYPE 7
0. QUIT
3
----- TYPE 3 -----
** Find all products sold in the past year. **
**Past year's sold products **
P_ID      company_name  sale_price
-----
1003-03,   Samsung,         100
2001-01,   Apple,           700
2003-01,   Apple,          2000
3001-03,   Sony,            300
3002-03,   Sony,            600
4004-03,   IRIVER,          10
----- Subtypes in TYPE 3 -----
1. TYPE 3-1.
2. TYPE 3-2.
-----
----- TYPE 3-1 -----
Input K? :
```

K를 입력하면 다음과 같이 표시된다.

```

C:\Users\DeskTop\Desktop\db\proj2\pj\64\Debug\pj.exe
(TYPE 3-1)top k products by dollar-amount sold
P_ID      company_name    sale_price
-----
2003-01,   Apple,          2000
2001-01,   Apple,              700
3002-03,   Sony,               600
3001-03,   Sony,               300

----- TYPE 3-2 -----
(TYPE 3-2) top 10 percent products by dollar-amount sold
P_ID      company_name    sale_price
-----
2003-01,   Apple,          2000

----- SELECT QUERY TYPES -----
1. TYPE 1
2. TYPE 2
3. TYPE 3
4. TYPE 4
5. TYPE 5
6. TYPE 6
7. TYPE 7
0. QUIT

```

4, 4-1, 4-2 는 Find all products by unit sales인데, 판매 단위로 상품을 나타내는 것이다. Sales를 만들 때, product_ID별로 하나의 상품만이 팔렸다고 저장했기 때문에 3, 3-1, 3-2와 동일한 쿼리를 가진다.

다음은 입력 화면이다.

C:\Users\DeskTop\Desktop\pdb\proj2\pj\wx64\Debug\pj.exe

3. TYPE 3
4. TYPE 4
5. TYPE 5
6. TYPE 6
7. TYPE 7
0. QUIT

4

----- TYPE 4 -----

** Find all products by unit sales in the past year. **

**Past year's sold products **

P_ID	company_name	sale_price
------	--------------	------------

1003-03,	Samsung,	100
2001-01,	Apple,	700
2003-01,	Apple,	2000
3001-03,	Sony,	300
3002-03,	Sony,	600
4004-03,	IRIVER,	10

----- Subtypes in TYPE 4 -----

1. TYPE 4-1.
2. TYPE 4-2.

----- TYPE 4-1 -----

Input K? :

C:\Users\DeskTop\Desktop\pdb\proj2\pj\wx64\Debug\pj.exe

P_ID	company_name	sale_price
------	--------------	------------

2003-01,	Apple,	2000
2001-01,	Apple,	700
3002-03,	Sony,	600
3001-03,	Sony,	300
1003-03,	Samsung,	100

----- TYPE 4-2 -----

(TYPE 4-2) top 10 percent products by dollar-amount sold

P_ID	company_name	sale_price
------	--------------	------------

2003-01,	Apple,	2000
----------	--------	------

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

1. TYPE 1
2. TYPE 2
3. TYPE 3
4. TYPE 4
5. TYPE 5
6. TYPE 6
7. TYPE 7
0. QUIT