데이터베이스 텀프로젝트 보고서

AI컴퓨터공학과 202111477 김지윤

**기말 과제 제안서 제출된 내용과 대비 차이점, 발전성**

1. team 테이블 삭제

이유: singer테이블에는 ‘대표곡’ 속성이 있는데 개인 멤버 하나하나를 singer로 등록할 때 대표곡에 null값이 들어가는 경우가 많아 team 테이블을 삭제하고 singer에 팀 단위로 등록하였다.

1. 소속가수가 음원과 발매 관계를 갖도록 수정, 음원 회사는 가수와 소속 관계만 가짐

이유: 가수가 음원에 대한 정보를 더 많이 가지고 있으므로 데이터 검색과 연산을 할 때 편리성을 높여준다.

1. 음원의 속성 ‘가수명’ 삭제

이유: 발매 관계를 갖는 개체들이 바뀌면서 두 개체에 같은 속성이 있으므로 음원 개체에서 삭제하였다. 필요시 조인을 통해 값을 얻을 수 있다.

1. 고객 테이블의 속성 ‘생년월일’을 ‘연령대’로 변경

이유: 월과 일의 값이 다른 값들에 비해 상대적으로 중요하지 않아 삭제함으로써 불필요한 메모리 낭비를 막았고 태어난 년도에서 연령대를 추출하는 방법이 아닌 입력 자체를 연령대로 하여 편리성을 높였다.

1. 고객 개체의 등급 속성 삭제

이유: 등급을 미리 정해두지 않고 listen 테이블과 조인한 후sql의 DML문을 이용하여 등급을 추출하는 것이 더 합리적이라 판단하였다.

1. 고객 개체의 선호 가수 속성 삭제

이유: 가수의 장르는 하나지만 하나의 장르를 노래하는 가수는 많기 때문에 선호 가수는 삭제해도 큰 차이가 없을 것이라고 판단하였다.

**요구사항 명세서**

• 음원 스트리밍 서비스에는 다양한 음원 회사들이 등록되어있다. 음원 회사는 회사명(고유함), 대표 이름, 회사 전화번호 정보를 갖고 있다.

• 음원 스트리밍 서비스에는 다양한 음원들이 등록되어있다. 음원은 음원 번호(고유함), 제목, 프로듀서 명, 가수 명, 순위 정보를 갖고 있다.

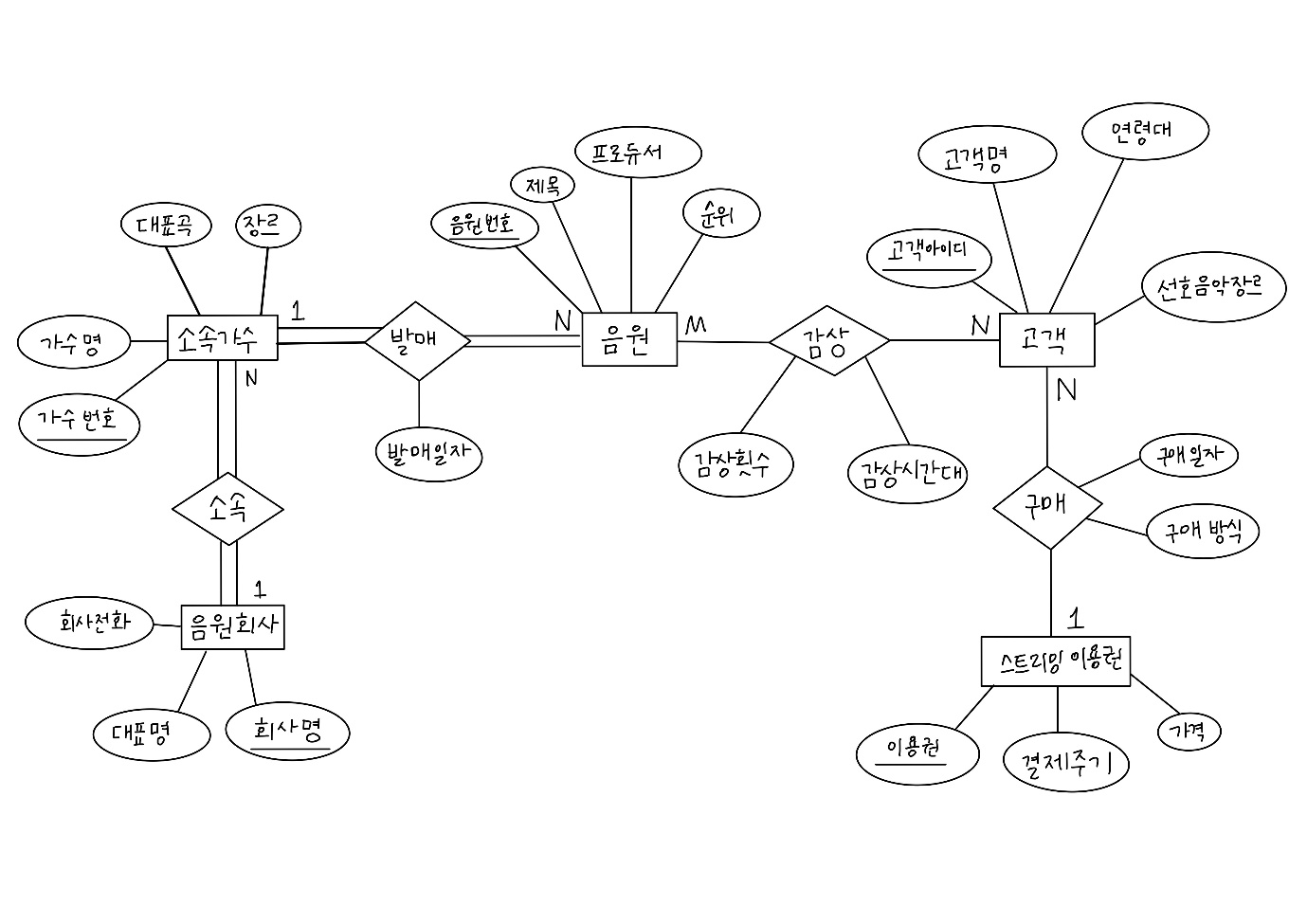
• 소속 가수는 음원을 발매한다. 한 명의 가수는 여러 개의 음원을 발매할 수 있지만 하나의 음원은 한 명의 가수만 발매할 수 있다. 가수가 음원을 발매하면 발매 일자가 자동으로 저장된다.

• 음원 회사에는 가수가 소속되어있다. 가수는 가수 번호(고유함), 가수 이름, 대표곡, 노래 장르 정보를 갖는다. 음원 회사는 여러 명의 가수를 소속할 수 있지만 가수는 하나의 회사에만 소속될 수 있다.

• 고객은 음원을 감상한다. 고객은 고객 아이디(고유함), 고개 이름, 연령대, 선호 음악 장르 정보를 갖는다. 고객이 음원을 감상하면 자동으로 감상 횟수와 감상 시간대가 저장된다. 모든 고객이 음원을 감상하지는 않으며 모든 노래가 고객에게 감상 되는 것도 아니다.

• 고객은 스트리밍 이용권을 구매하여 스트리밍 서비스를 이용할 수 있다. 구매 시 구매 일자와 구매 방식이 자동으로 저장된다. 스트리밍 이용권은 이용권 이름(고유함), 결제 주기, 가격 정보를 갖고 있다. 스트리밍 이용권은 여러 명의 고객에 의해 구매될 수 있지만 한 명의 고객은 하나의 스트리밍 이용권만 구매할 수 있다. 단, 모든 고객이 스트리밍 이용권을 구매하지는 않는다.

**ER 다이어그램**

****

**테이블 정의서**

* enter(음원회사) 테이블

create table enter(

회사명 varchar2(30) primary key,

대표명 varchar2(20),

회사전화 number(10)

);

회사명을 기본키로 설정하였다. 대표와 전화는 복수이거나 없을 수 있으므로 기본키가 아니다.

* singer(소속가수) 테이블

create table singer(

가수번호 number(4) primary key,

가수명 varchar2(30) not null,

대표곡 varchar2(30) not null,

장르 varchar(30) not null,

회사명 varchar2(30),

foreign key (회사명) references enter(회사명)

);

가수번호를 기본키로 설정한다. enter 테이블과 소속 관계를 가지므로 기본키인 회사명을 외래키로 설정한다.

* song(음원) 테이블

create table song(

음원번호 number(4) primary key,

제목 varchar2(30) not null,

프로듀서 varchar2(30),

순위 number(3) check(순위>=1 and 순위<=100),

가수번호 number(4),

발매일자 varchar2(10),

foreign key (가수번호) references singer(가수번호)

);

음원번호를 기본키로 설정하였다. 제목은 중복이 있을 수 있기 때문에 부적합하다. singer 테이블과 발매 관계를 가지므로 기본키를 외래키로 설정한다. 노래의 순위는 100위까지 나타내므로 check를 통해 순위 속성을 1이상 100이하로 제약한다.

* streaming\_pass(이용권) 테이블

create table streaming\_pass(

이용권 varchar2(30) primary key,

결제주기 number(2),

가격 number(10)

);

이용권 이름을 기본키로 설정한다.

* customer(고객) 테이블

create table customer(

고객아이디 varchar2(30) primary key,

고객명 varchar2(30),

연령대 varchar2(10),

등급 varchar2(10),

선호음악장르 varchar2(30),

이용권 varchar2(30),

구매일자 varchar2(10),

구매방식 varchar2(30),

foreign key(이용권) REFERENCES streaming\_pass(이용권)

);

고객 아이디를 기본키로 설정한다. streaming\_pass 테이블과 구매 관계를 가지므로 기본키를 외래키로 설정한다.

* listen(감상) 테이블

create table listen(

음원번호 number(4),

고객아이디 varchar2(30),

감상횟수 number(4) check(감상횟수>=0 and 감상횟수<=1000),

감상시간대 varchar(20),

foreign key(음원번호) references song(음원번호),

foreign key(고객아이디) references customer(고객아이디),

constraint listen\_pk primary key(음원번호,고객아이디)

);

listen 테이블은 song 테이블과 customer 테이블의 관계 속성에서 나온 것으로 외래키 두 개를 기본키로 설정한다. 한 고객은 하나의 노래를 1000회 초과로 들을 수 없다.

**SQL문 실행**

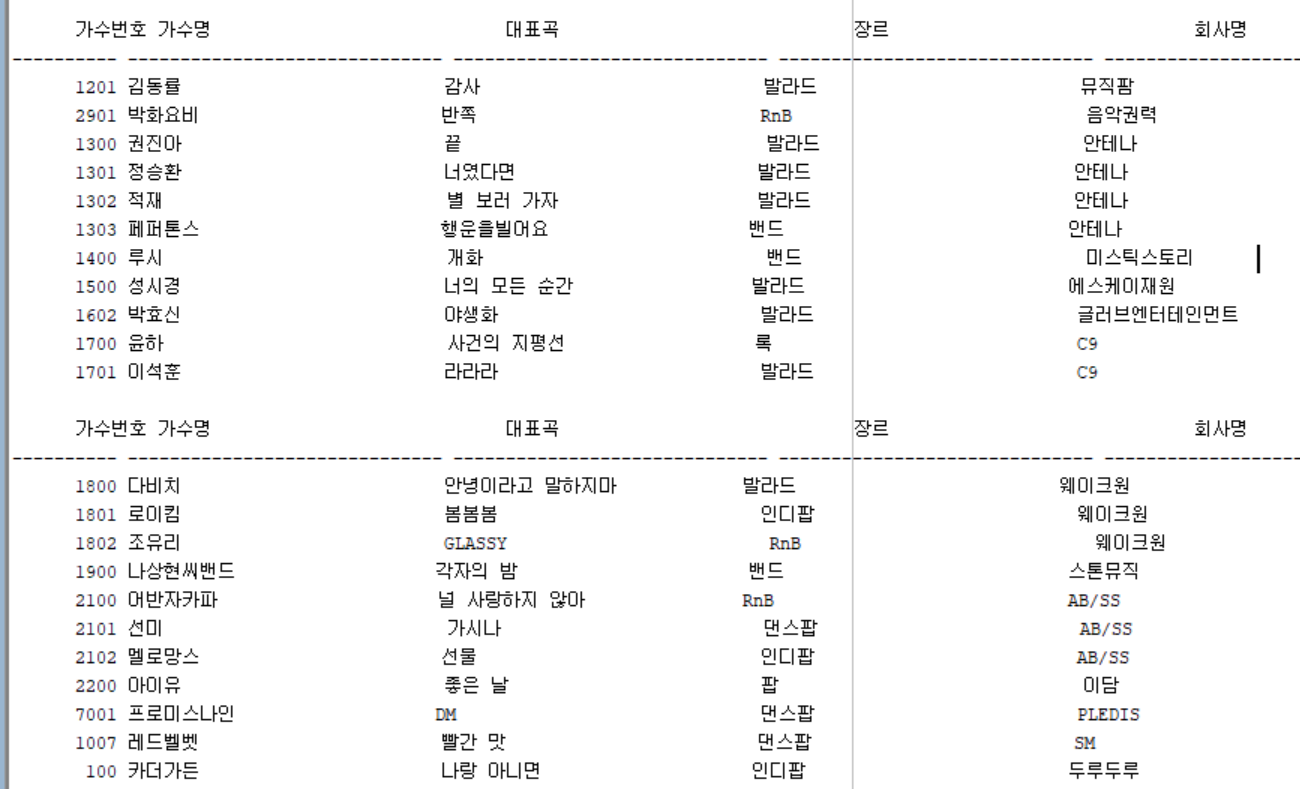
1. INSERT : 각각의 테이블에 데이터를 입력한다.
2. ENTER 테이블 결과

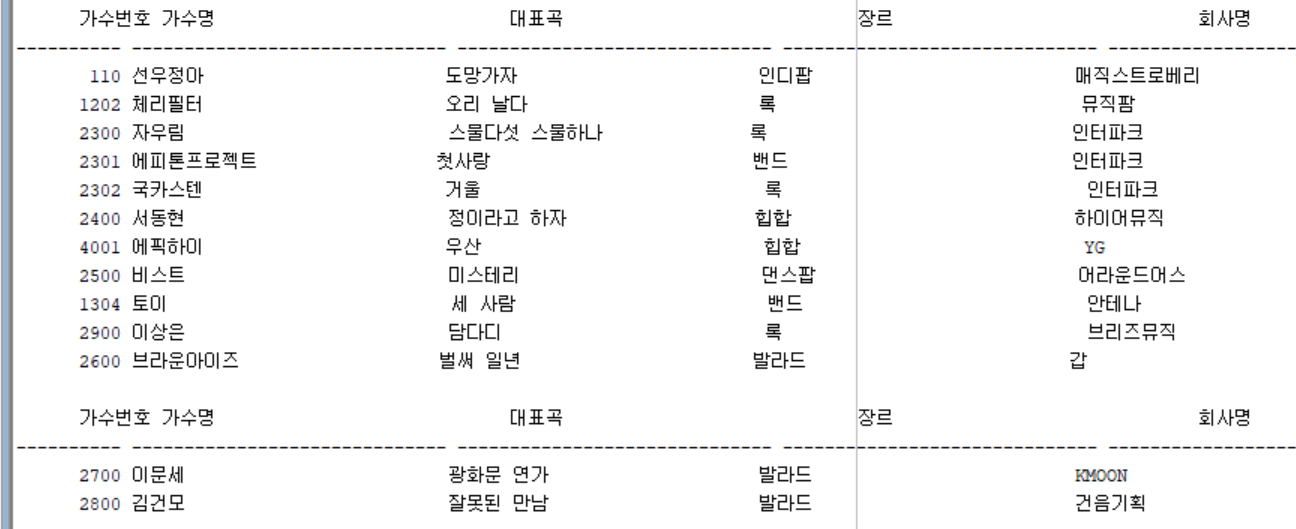




1. SINGER 테이블 결과

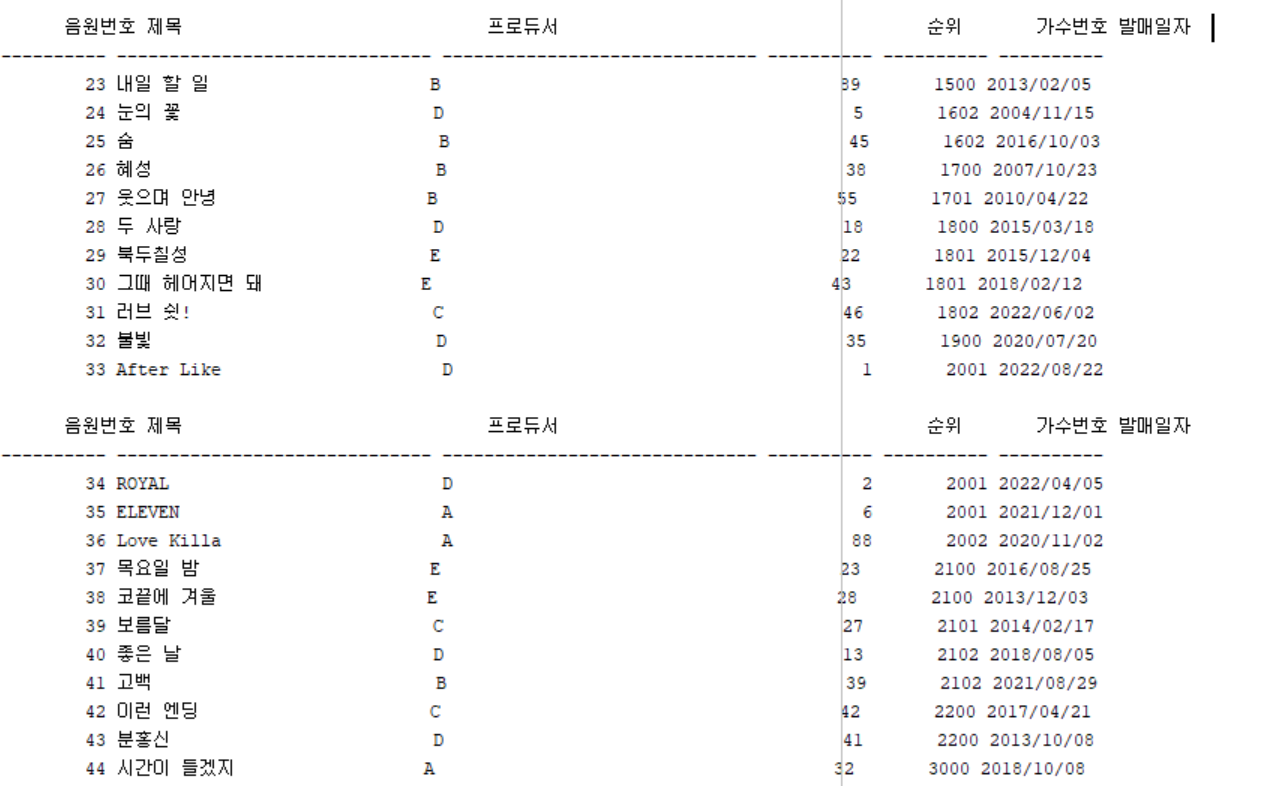


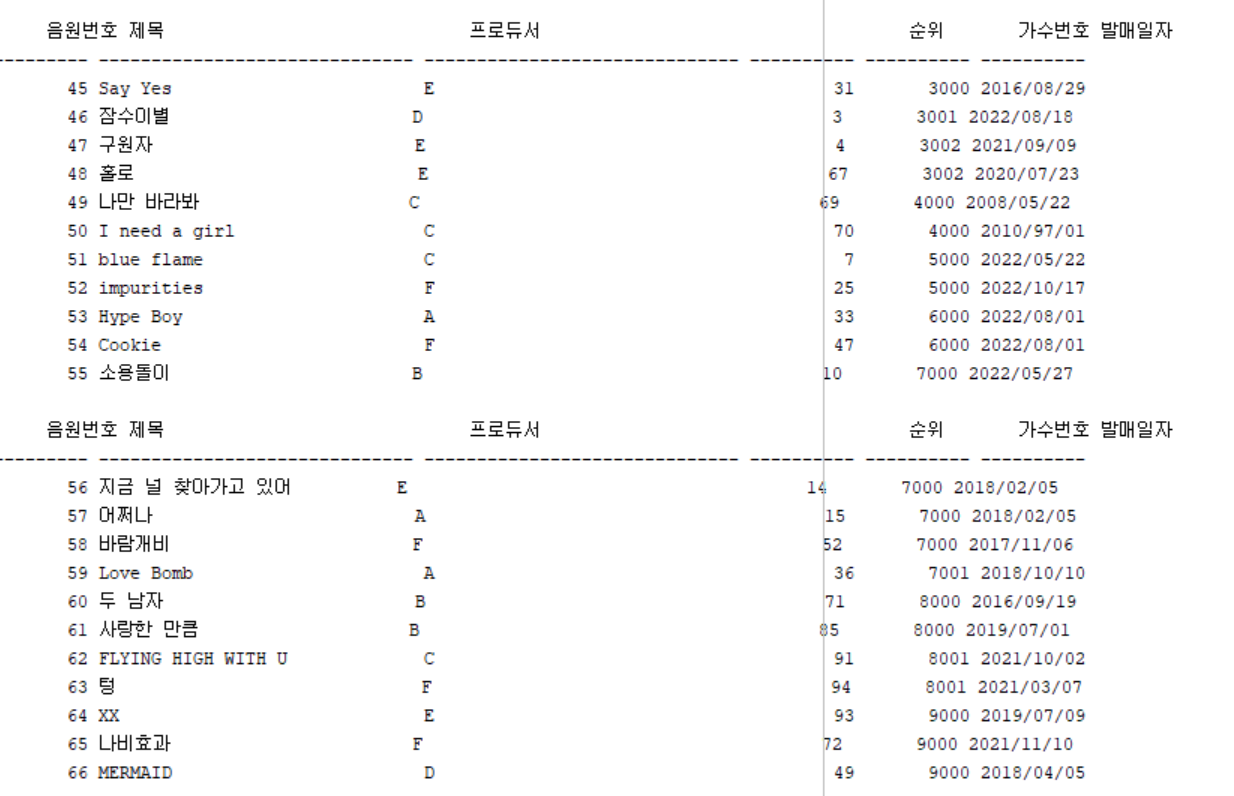


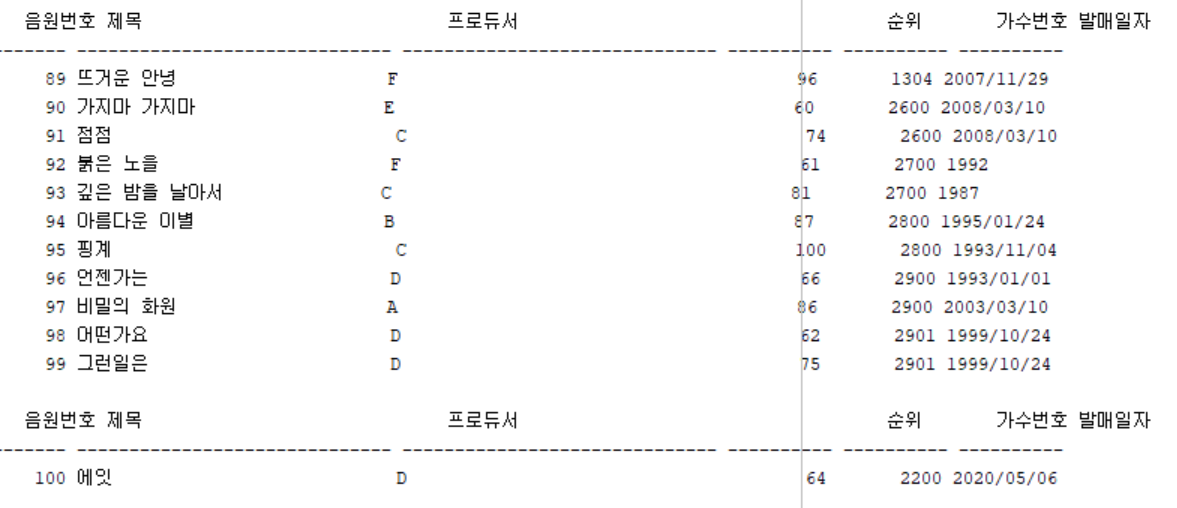
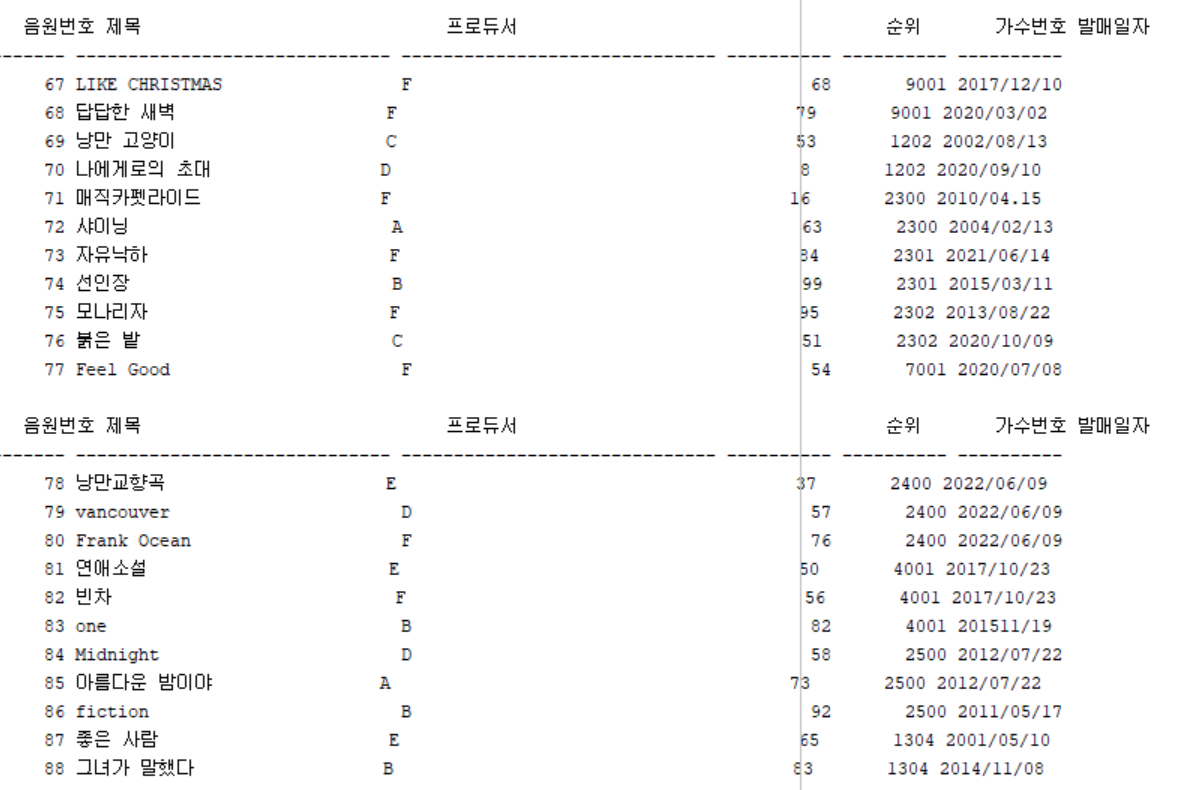


1. SONG 테이블 결과

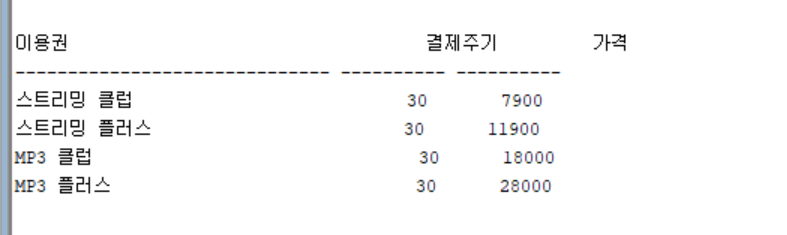






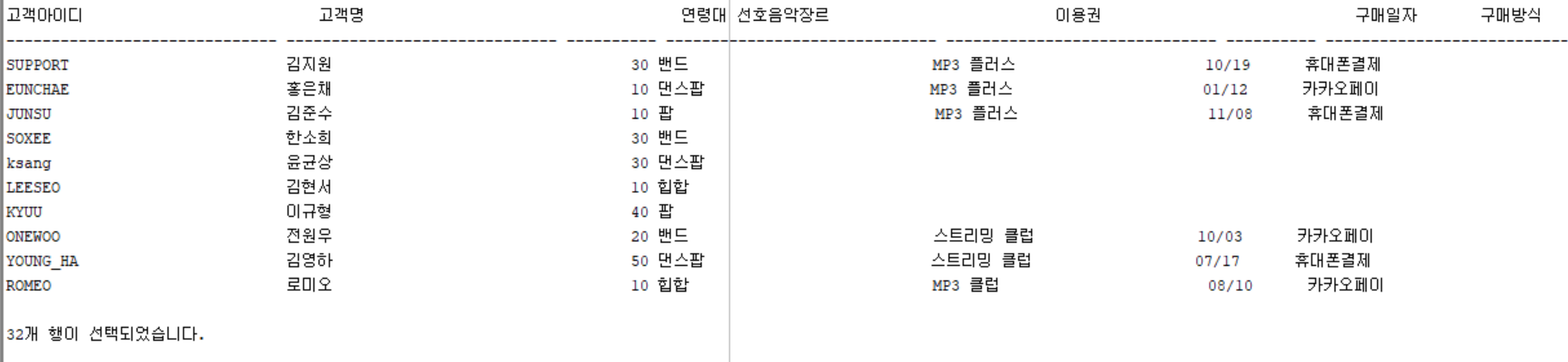


1. STREAMING\_TABLE 테이블 결과

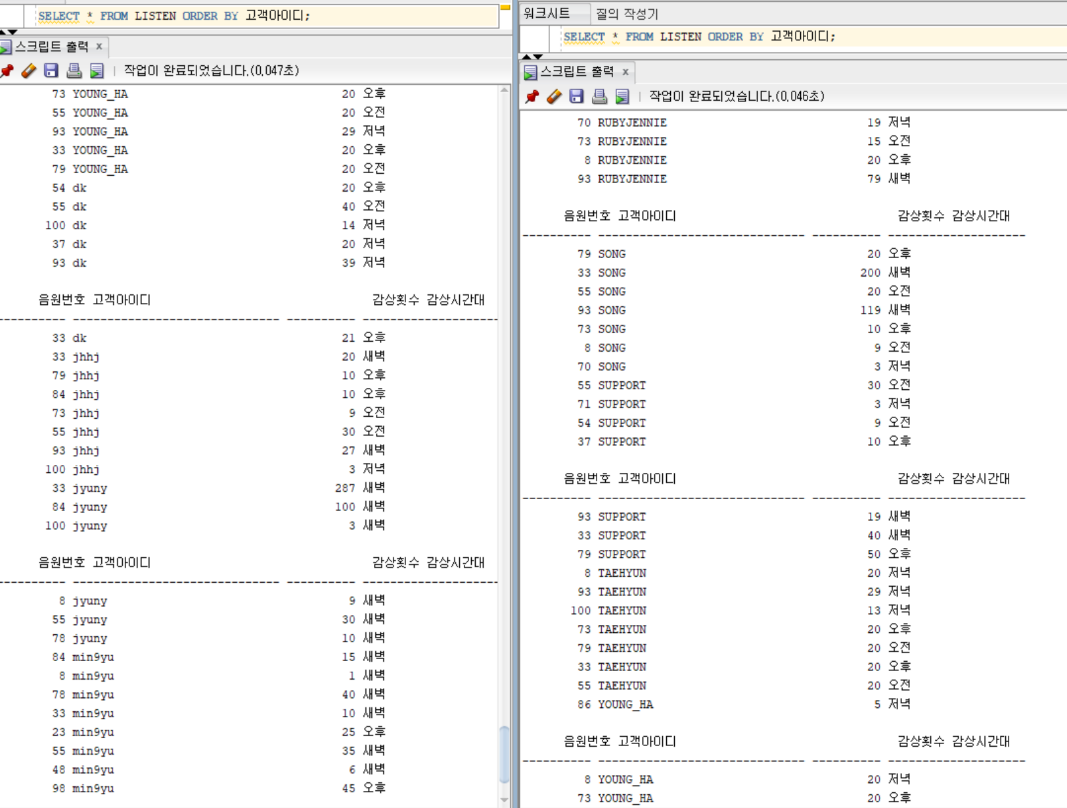


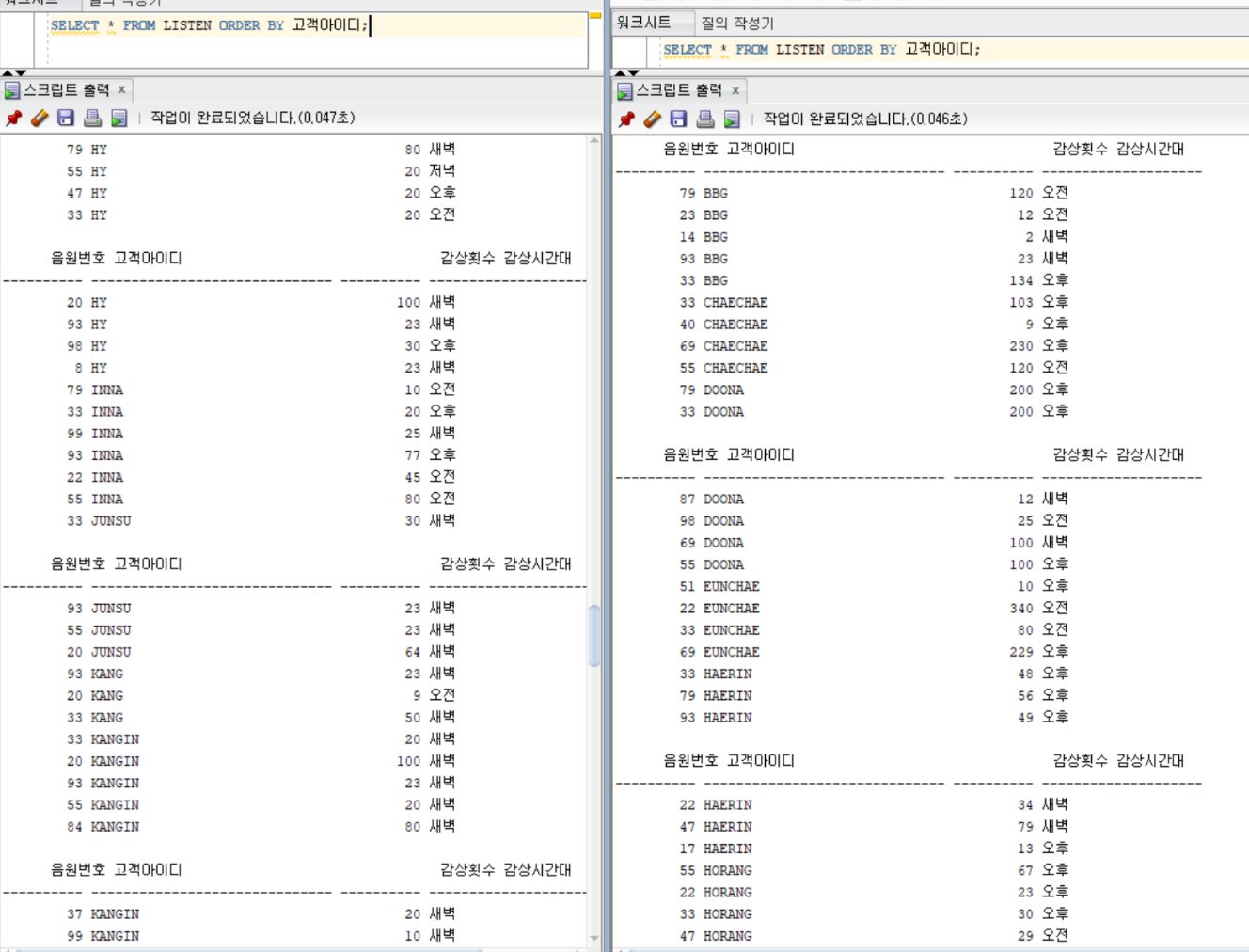
1. CUSTOMER 테이블 결과

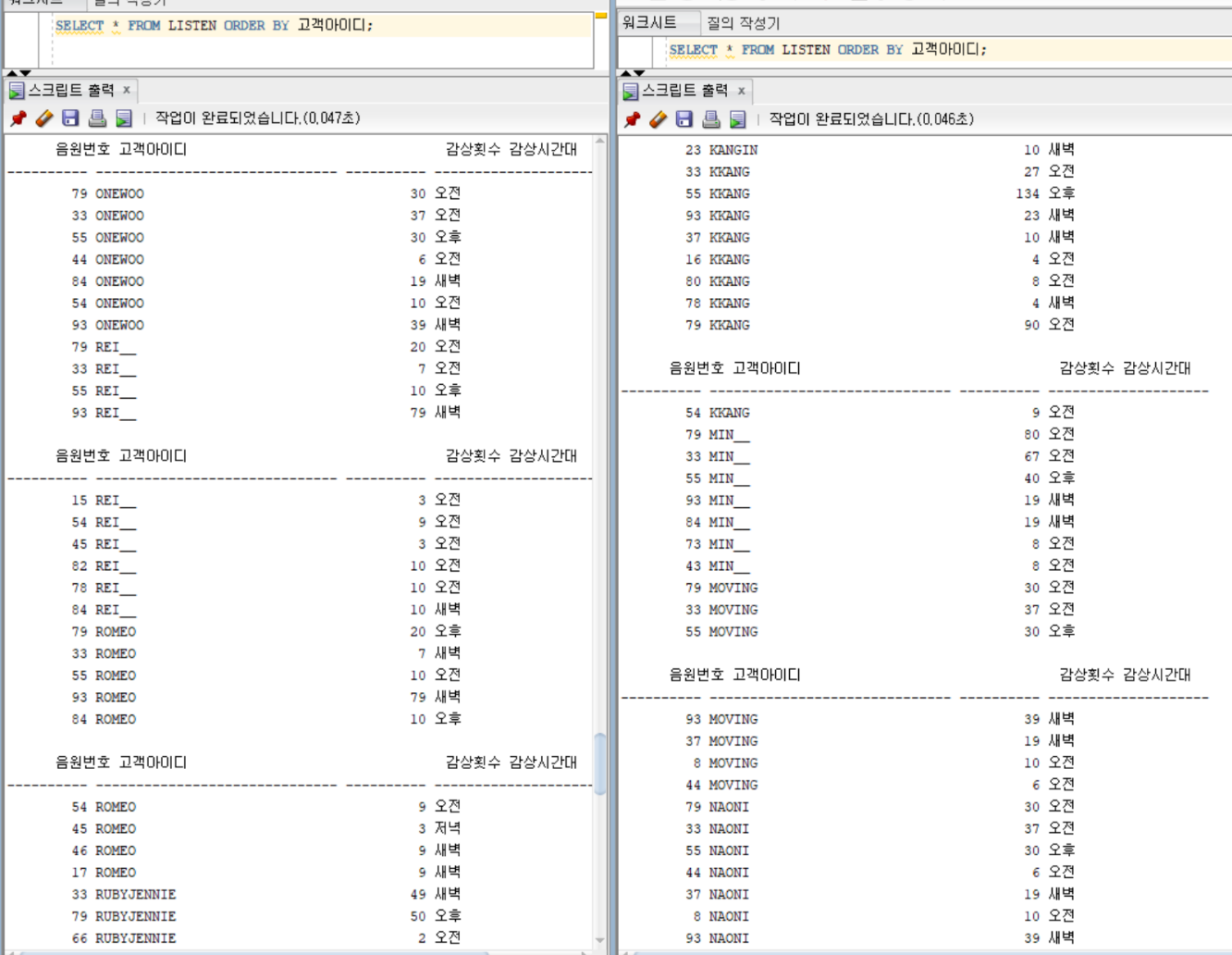




1. LISTEN 테이블 결과

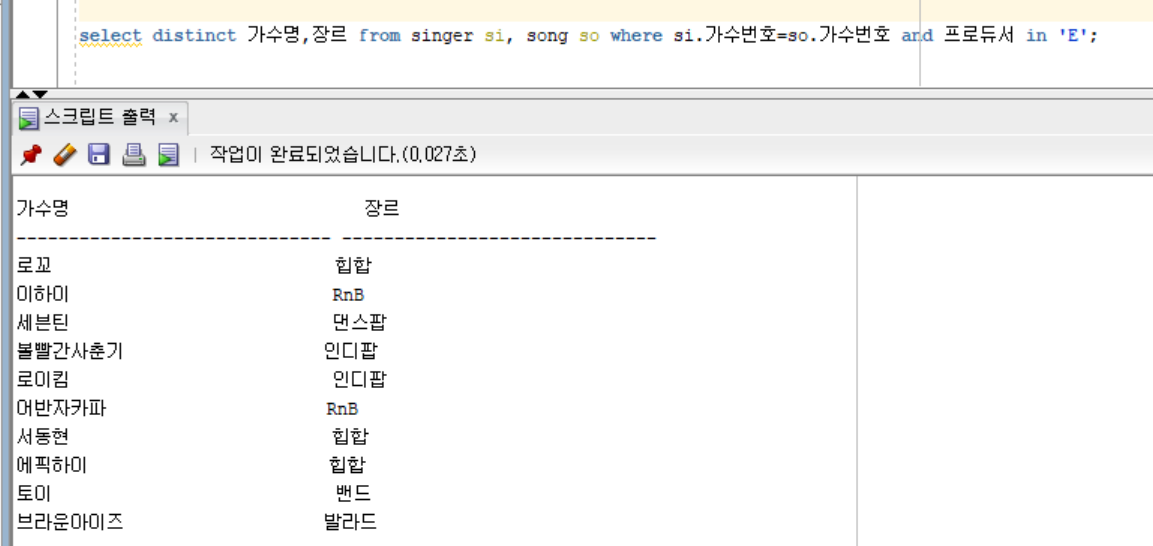






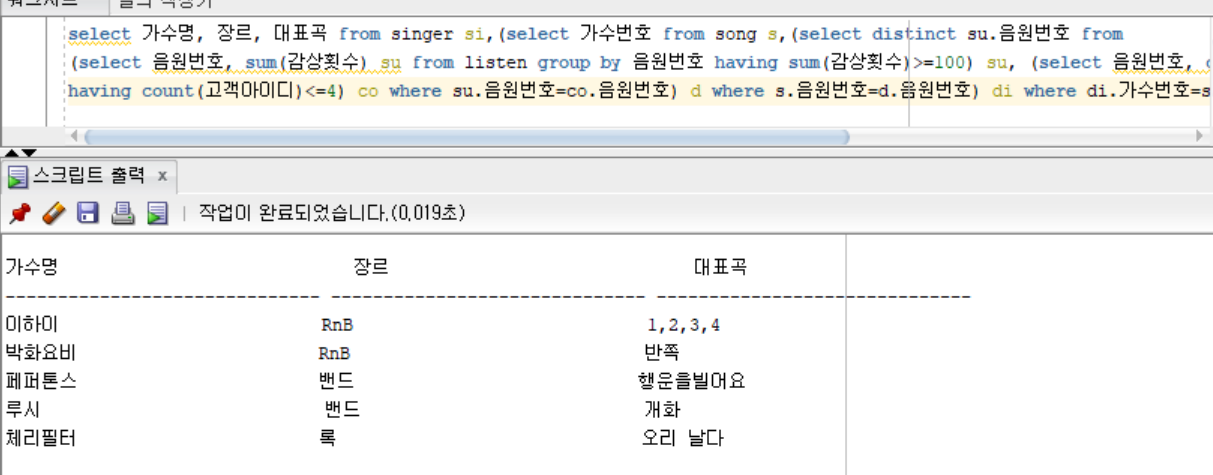
1. 여러가지 SQL문

* E프로듀서가 프로듀싱한 가수와 그 장르
* select distinct 가수명,장르 from singer si, song so where si.가수번호=so.가수번호 and 프로듀서 in 'E';



* 감상한 이용자는 얼마 없지만 감상횟수가 많은 노래 즉, 마니아층이 있는 노래의 가수와 장르 그 대표곡
* select 가수명, 장르, 대표곡 from singer si, (select 가수번호 from song s,

(select distinct su.음원번호 from (select 음원번호, sum(감상횟수) su from listen group by 음원번호having sum(감상횟수)>=100) su, (select 음원번호, count(고객아이디) co from listen group by 음원번호 having count(고객아이디)<=4) co where su.음원번호=co.음원번호) d where s.음원번호=d.음원번호) di where di.가수번호=si.가수번호;



* 프로듀서가 연도별로 프로듀싱한 노래 수
* select 프로듀서,

sum(decode(substr(so.발매일자,1,3),'198',1,0 )) in\_1980,

sum(decode(substr(so.발매일자,1,3),'199',1,0 )) in\_1990,

sum(decode(substr(so.발매일자,1,3),'200',1,0 )) in\_2000,

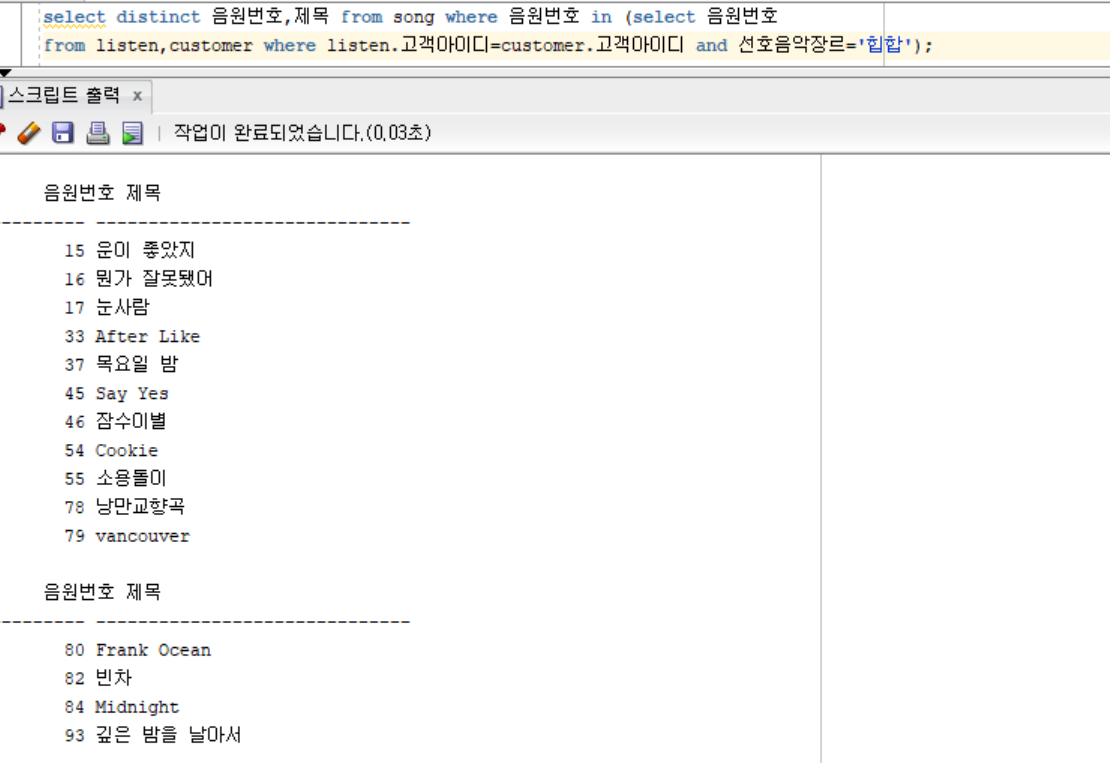
sum(decode(substr(so.발매일자,1,3),'201',1,0 )) in\_2010,

sum(decode(substr(so.발매일자,1,3),'202',1,0 )) in\_2020

from song so group by 프로듀서 order by 프로듀서



* 선호 음악 장르가 같은 사람이 듣는 음악 ex) 힙합일 때
* select distinct 음원번호,제목 from song where 음원번호 in (select 음원번호 from listen,customer where listen.고객아이디=customer.고객아이디 and 선호음악장르='힙합');



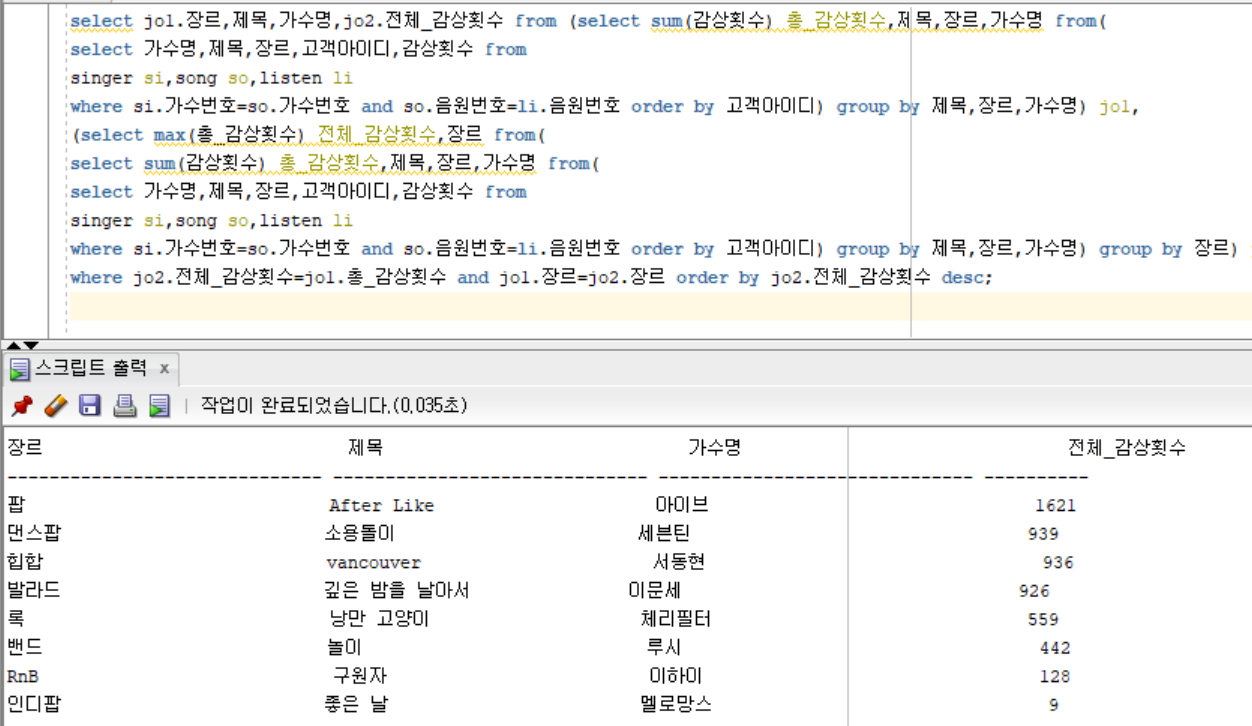
* 장르별 음원의 최대 감상횟수와 그 음원의 제목
* select jo1.장르,제목,가수명,jo2.전체\_감상횟수 from (select sum(감상횟수) 총\_감상횟수,제목,장르,가수명 from( select 가수명,제목,장르,고객아이디,감상횟수 from

singer si,song so,listen li where si.가수번호=so.가수번호 and so.음원번호=li.음원번호 order by 고객아이디) group by 제목,장르,가수명) jo1,

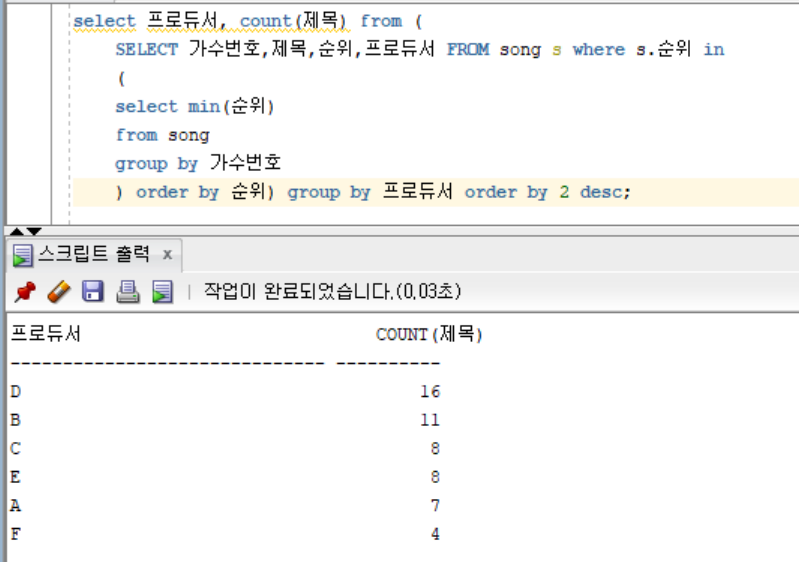
(select max(총\_감상횟수) 전체\_감상횟수,장르 from( select sum(감상횟수) 총\_감상횟수,제목,장르,가수명 from( select 가수명,제목,장르,고객아이디,감상횟수 from

singer si,song so,listen li where si.가수번호=so.가수번호 and so.음원번호=li.음원번호 order by 고객아이디) group by 제목,장르,가수명)

group by 장르) jo2 where jo2.전체\_감상횟수=jo1.총\_감상횟수 and jo1.장르=jo2.장르 order by jo2.전체\_감상횟수 desc;



* 가수별 음원 성적이 가장 높은 음원을 작곡한 가수와 작곡한 음원 수
* select 프로듀서, count(제목) from ( SELECT 가수번호,제목,순위,프로듀서 FROM song s where s.순위 in ( select min(순위) from song group by 가수번호) order by 순위) group by 프로듀서 order by 2;



* 고객별 총 음원 감상 횟수와 그에 따른 등급, 보상 (단, 총 감상 횟수가 500 이상이면 MVIP, 400 이상이면 VIP, 300 이상이면 GOLD, 200 이상이면 SILVER, 100 이상이면 USER이다)
* SELECT 고객아이디, 총\_스트리밍\_수,

CASE

WHEN 총\_스트리밍\_수 >=500 THEN 'MVIP'

WHEN 총\_스트리밍\_수 >=400 THEN 'VIP'

WHEN 총\_스트리밍\_수 >=300 THEN 'GOLD'

WHEN 총\_스트리밍\_수 >=200 THEN 'SILVER'

ELSE 'USER'

END AS 등급,

CASE

WHEN 총\_스트리밍\_수 >=500 THEN '이용권\_업그레이드'

WHEN 총\_스트리밍\_수 >=400 THEN '이용권\_무료연장\_2개월'

WHEN 총\_스트리밍\_수 >=300 THEN 'MP3\_이용\_20회'

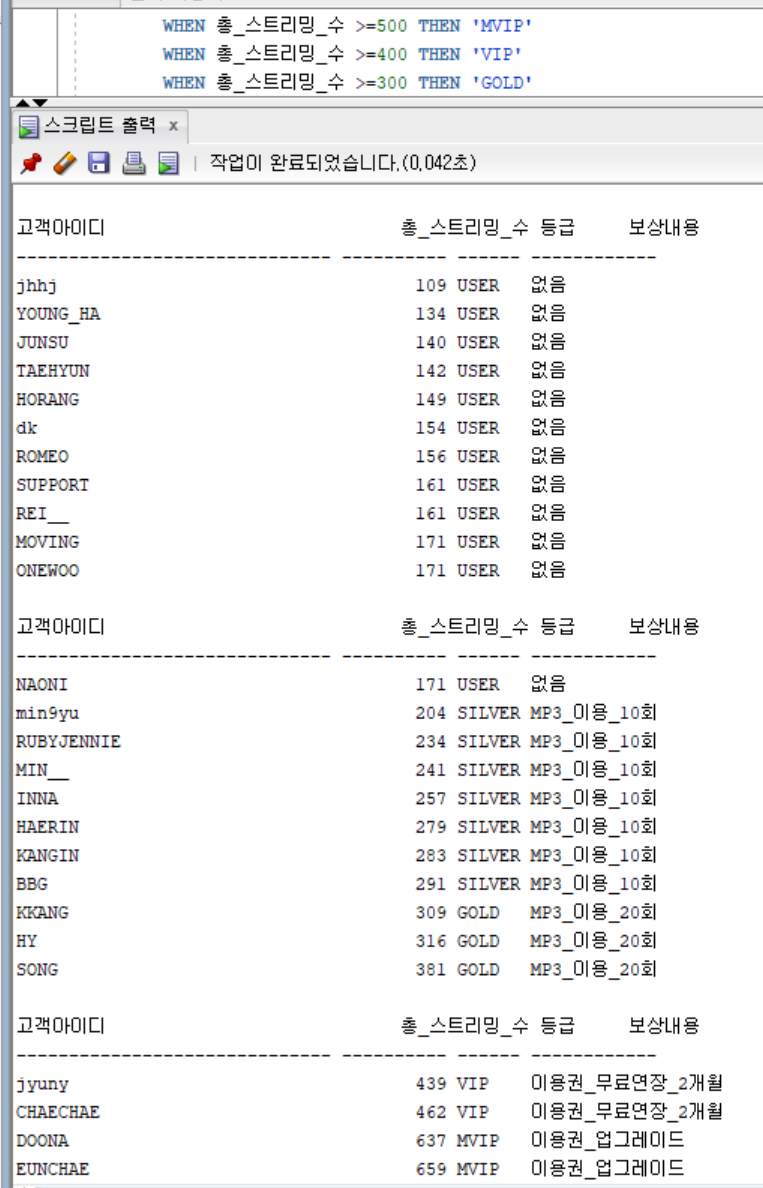
WHEN 총\_스트리밍\_수 >=200 THEN 'MP3\_이용\_10회'

ELSE '없음'

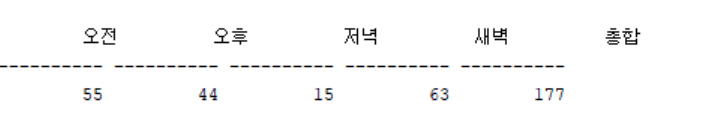
END AS 보상내용

FROM (select 고객아이디,sum(감상횟수) 총\_스트리밍\_수 from listen group by 고객아이디)

WHERE 총\_스트리밍\_수>=100 ORDER BY 총\_스트리밍\_수;

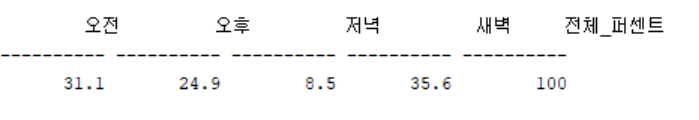


* 음원 스트리밍 시스템 이용자들이 가장 많이 이용한 시간대 (개수)
* select sum(DECODE(l.감상시간대,'오전',1,0)) 오전, sum(DECODE(l.감상시간대,'오후',1,0)) 오후, sum(DECODE(l.감상시간대,'저녁',1,0)) 저녁, sum(DECODE(l.감상시간대,'새벽',1,0)) 새벽, count(\*) 총합 FROM LISTEN l;



이를 비율로 바꿔서 출력하면,

* select round(sum(DECODE(l.감상시간대,'오전',1,0))/count(\*)\*100,1) 오전, round(sum(DECODE(l.감상시간대,'오후',1,0))/count(\*)\*100,1) 오후, round(sum(DECODE(l.감상시간대,'저녁',1,0))/count(\*)\*100,1) 저녁, round(sum(DECODE(l.감상시간대,'새벽',1,0))/count(\*)\*100,1) 새벽, 100 전체\_퍼센트 FROM LISTEN l;



* 스트리밍 클럽, MP3 클럽의 20% 가격 인상, 카카오페이 사용자의 경우 10% 인상
* select 고객명,st.이용권,가격, case when cu.이용권 like '%클럽' and 구매방식!='카카오페이' then 가격\*1.2 when cu.이용권 like '%클럽' and 구매방식='카카오페이' then 가격\*1.1 else 가격 end as 인상from streaming\_pass st,customer cu where st.이용권=cu.이용권 order by 인상

