

# 2차 미니 프로젝트

## 연습문제

case study 비디오 관리 대여점 시스템을 구축한다고 가정하여,  
필요한 객체(테이블 등)를 설계하고 분담하여 필요한 모든 쿼리문을 작성하세요.

1. 고객이 비디오를 대여 또는 반납하는 작업이 주업무이다.
2. 클라이언트는 고객에 대한 고객명, 나이, 주소, 전화번호 등의 정보관리를 원한다.
3. 클라이언트는 비디오에 대한 제목,장르,대여료,관람등급,출시사,출시일,대여정보 등의 관리를 원한다.
4. 특정 고객이 비디오를 빌려간 경우 비디오 테이프의 반납 예정 일자는 대여일로부터 보통 2일 (장르에 따라 달라질 수 있다)로 계산하고, 대여 상태를 기록하게 된다. 연체되었을 경우는 벌금을 하루에 200원씩 (장르에 따라 다를 수 있다) 부과한다. 최종 기본 대여료는 2000원 (장르에 따라 다를 수 있음) + 벌금으로 대여 총액을 계산하고 매출로 집계된다. 비디오가 반납되고 회수되었음을 기록하고, 대여 총액을 기록한다.
5. 비디오 관리 시스템 개발의 주된 목적은 비디오의 대여 반납을 주 업무로 하여 기간별 매출 총액 등을 (클라이언트가)확인할 수 있도록 하되, 고객관리와 비디오 관리를 부업무로 할 수 있도록 한다.

### 쿼리문

- 첫 방문 고객이 와서 회원가입한다.
- `insert into member values(...);`
- 손님이 아바타2 비디오가 있는지 문의한다.
- 클라이언트(사장님)가 0월 0일까지의 매출을 보고싶어 한다.
- 클라이언트가 장르별로 기본 대여료를 올리려고 한다.
- 필수 : 각 테이블의 CRUD 쿼리문  
팀별로 공학적 ERD 1개만 (공통)  
테이블 명세서 각 테이블 1개씩만 (공통)
- 옵션 : 팀원이 2~3문제씩 만들어서 이 문제를 해결하는 쿼리문을 작성

김준범

문제 1. 2020년부터 2021년 각 회원의 매출을 조회하고 반납날짜순으로 정렬해라 문제 1개

```

select member_id, sum(total_fee) as 매출, actual_return_date from rent_info
where rent_date between "2020-01-01"and"2021-12-31"
group by member_id
order by member_id;

```

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```

1  -- 문제 1. 2020년부터 2021년 각 회원의 매출을 조회하고 반납날짜순으로 정렬해라 문제 1개
2  • select member_id, sum(total_fee) as 매출, actual_return_date from rent_info
3    where rent_date between "2020-01-01"and"2021-12-31"
4    group by member_id
5    order by member_id;

```

The result grid displays the following data:

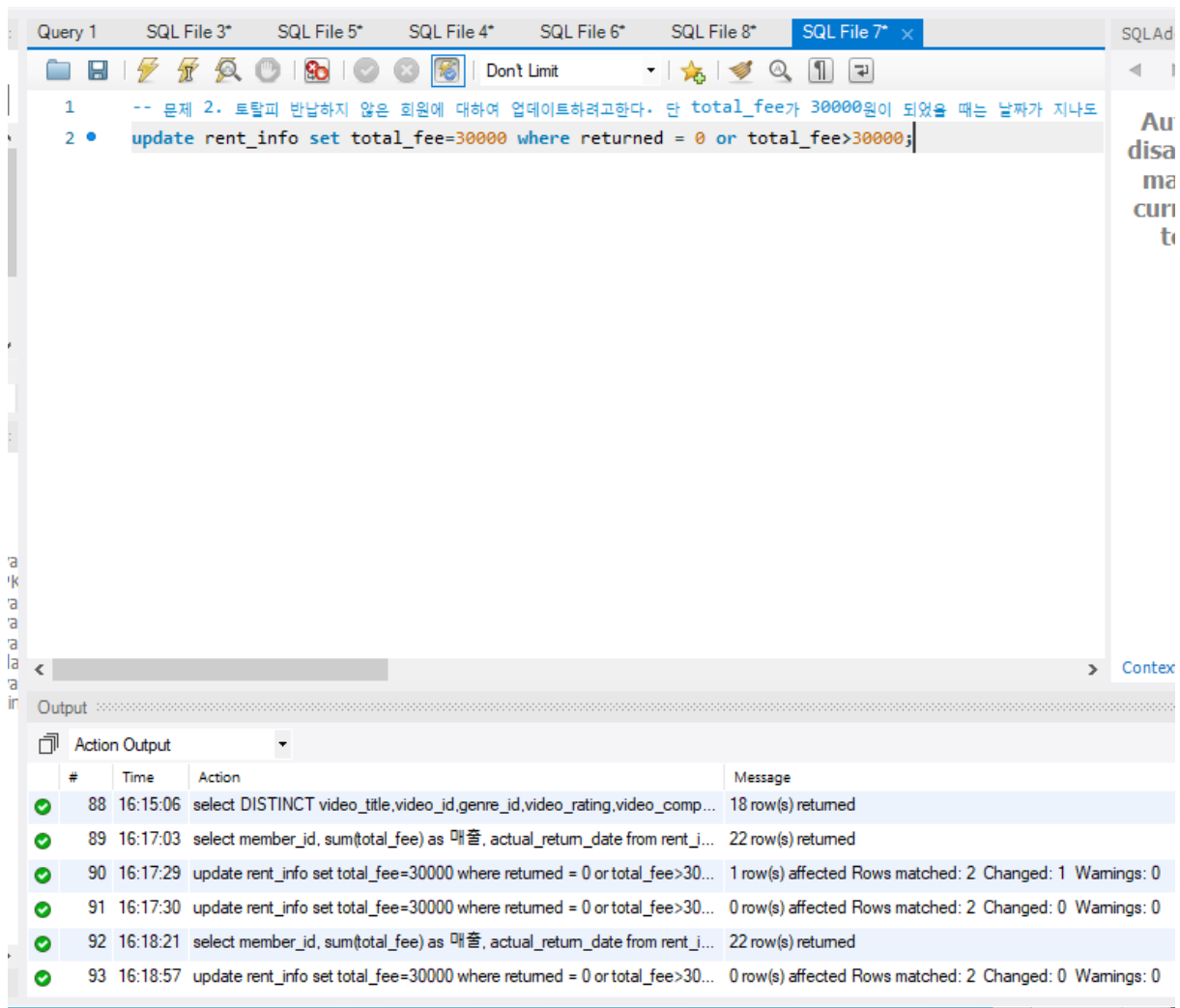
member_id	매출	actual_return_date
alice2023	2000	2020-01-06
amy_hill	30000	2021-02-05
bobsmith	2000	2020-02-19
brianscott	2000	2021-07-02
cara_green	5200	2020-03-25

문제 2. 토탈피 반납하지 않은 회원에 대하여 업데이트하려고한다. 단 total\_fee가 30000원이 되었을 때는 날짜가 지나도 토탈피가 오르지 않도록 하는 update문을 작성하세요.

```

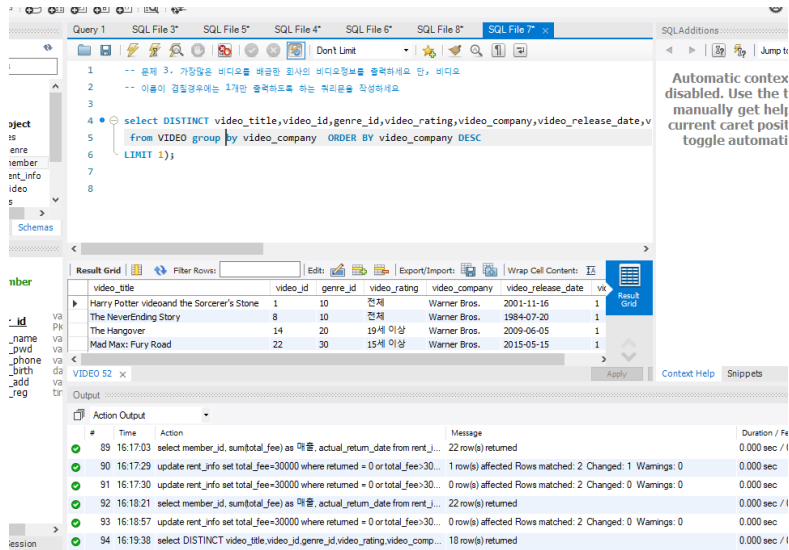
update rent_info set total_fee=30000 where returned = 0 or total_fee>30000;

```



문제 3. 가장많은 비디오를 배급한 회사의 비디오정보를 출력하세요 단, 비디오 이름이 겹칠 경우에는 1개만 출력하도록 하는 쿼리문을 작성하세요

```
select DISTINCT
video_title,video_id,genre_id,video_rating,video_company,video_release_date,video_reg
from VIDEO where video_company = (select video_company
from VIDEO group by video_company ORDER BY video_company DESC
LIMIT 1);
```



강진구

-- 문제1

-- 사장님이 이벤트를 위해 출시일이 30년이 넘은 비디오를 조회하려 한다.(비디오 정보조회, 총 개수 조회 각각 쿼리문)

select \* from video;

-- 1-1. 비디오 정보 조회

select video\_id, video\_title from video where video\_release\_date < date\_sub(now(), INTERVAL 30 YEAR);

-- 1-2. 개수 조회

select count(\*) from video where video\_release\_date < date\_sub(now(), INTERVAL 30 YEAR);

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations and a 'Don't Limit' dropdown. The query window contains the following SQL code:

```

1  -- 문제1 사장님이 이벤트를 위해 출시일이 30년이 넘은 비디오를 조회하려 한다. (비디오 정보조회, 총 개수 조회 각각 쿼리문)
2  • select * from video;
3  -- 1-1. 비디오 정보 조회
4  • select video_id, video_title from video where video_release_date < date_sub(now(), INTERVAL 30 YEAR);
5  -- 1-2. 개수 조회
6  • select count(*) from video where video_release_date < date_sub(now(), INTERVAL 30 YEAR);

```

The 'Result Grid' shows the results of the first query (line 4):

video_id	video_title
7	The Princess Bride
8	The NeverEnding Story
10	Willow
16	Ferris Bueller's Day Off
19	Airplane!

The 'Output' window shows the execution log:

#	Time	Action	Message
93	16:18:57	update rent_info set total_fee=30000 where returned = 0 or total_fee>30...	0 row(s) affected Rows matched: 2 Changed: 0 Warnings: 0
94	16:19:38	select DISTINCT video_title,video_id,genre_id,video_rating,video_comp...	18 row(s) returned
95	16:20:20	select * from video	100 row(s) returned
96	16:20:21	select video_id, video_title from video where video_release_date < date...	22 row(s) returned
97	16:20:22	select count(*) from video where video_release_date < date_sub(now(), I...	1 row(s) returned
98	16:20:24	select video_id, video_title from video where video_release_date < date...	22 row(s) returned

- 문제2
    - 오픈 이후 세 번 이상 대여된 비디오의 정보를 횡수 기준으로 내림차순으로 정렬하여 출력하라. (video\_id, video\_title, 총 대여 횡수 포함)
- ```

select v.video_id, v.video_title, count(*) as count
from rent_info r
join video v on r.video_id = v.video_id
group by v.video_id
having count >= 3
order by count desc;

```

Query 1   SQL File 3\*   SQL File 5\*   SQL File 4\*   SQL File 6\*   SQL File 8\*   SQL File 7\* x

1 -- 문제2 오픈 이후 세 번 이상 대여된 비디오의 정보를 횡수 기준으로 내림차순으로 정렬하여 출력하라. (video\_id, vid  
 2 • select v.video\_id, v.video\_title, count(\*) as count  
 3 from rent\_info r  
 4 join video v on r.video\_id = v.video\_id  
 5 group by v.video\_id  
 6 having count >= 3  
 7 order by count desc;  
 8

Result Grid

|   | video_id | video_title                           | count |
|---|----------|---------------------------------------|-------|
| ▶ | 45       | Gone Girl                             | 7     |
|   | 67       | The Fifth Element                     | 6     |
|   | 12       | Anchorman: The Legend of Ron Burgundy | 5     |
|   | 78       | The Incredibles                       | 5     |
|   | 89       | West Side Story                       | 5     |

Result 57 x   Read Or

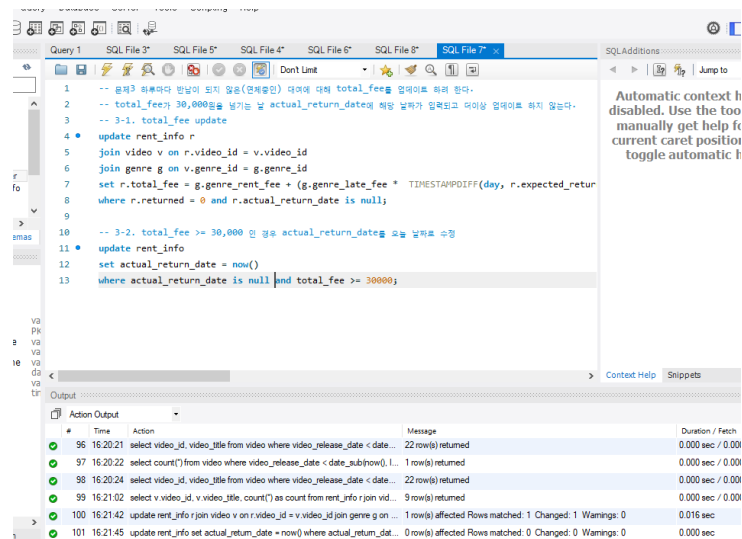
Output

Action Output

| #    | Time     | Action                                                                           | Message             |
|------|----------|----------------------------------------------------------------------------------|---------------------|
| ✓ 94 | 16:19:38 | select DISTINCT video_title,video_id,genre_id,video_rating,video_comp...         | 18 row(s) returned  |
| ✓ 95 | 16:20:20 | select * from video                                                              | 100 row(s) returned |
| ✓ 96 | 16:20:21 | select video_id, video_title from video where video_release_date < date...       | 22 row(s) returned  |
| ✓ 97 | 16:20:22 | select count(*) from video where video_release_date < date_sub(now(), l...       | 1 row(s) returned   |
| ✓ 98 | 16:20:24 | select video_id, video_title from video where video_release_date < date...       | 22 row(s) returned  |
| ✓ 99 | 16:21:02 | select v.video_id, v.video_title, count(*) as count from rent_info r join vid... | 9 row(s) returned   |

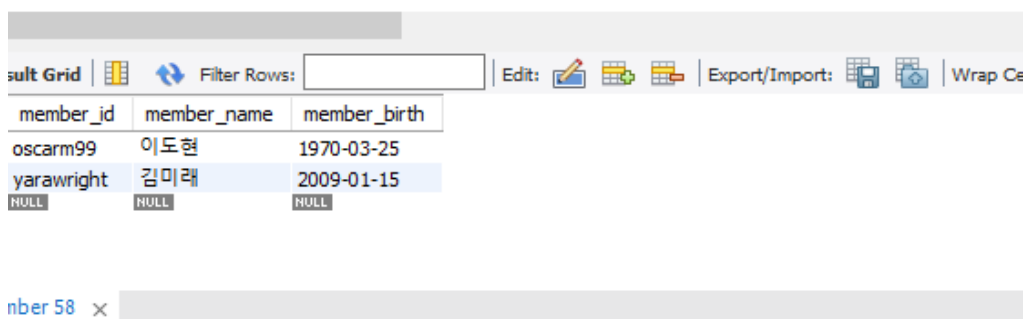
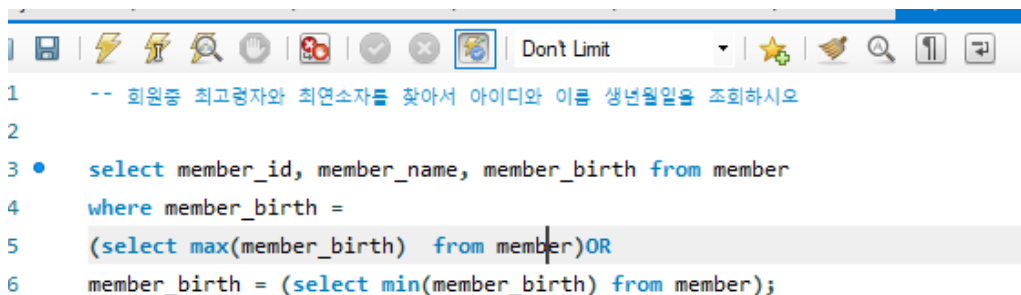
- 문제3
  - 하루마다 반납이 되지 않은(연체중인) 대여에 대해 total\_fee를 업데이트 하려 한다.
  - total\_fee가 30,000원을 넘기는 날 actual\_return\_date에 해당 날짜가 입력되고 더이상 업  
데이트 하지 않는다.
  - 3-1. total\_fee update
 

```
update rent_info r
join video v on r.video_id = v.video_id
join genre g on v.genre_id = g.genre_id
set r.total_fee = g.genre_rent_fee + (g.genre_late_fee * TIMESTAMPDIF(day,
r.expected_return_date, now()))
where r.returned = 0 and r.actual_return_date is null;
```
  - 3-2. total\_fee >= 30,000 인 경우
    - actual\_return\_date를 오늘 날짜로 수정
    - update rent\_info
    - set actual\_return\_date = now()
    - where actual\_return\_date is null and total\_fee >= 30000;



## 연화

- 문제 1회원중 최고령자와 최연소자를 찾아서 아이디와 이름 생년월일을 조회하시오
- `select member_id, member_name, member_birth`  
`from member`  
`where member_birth = (select max(member_birth) from member)OR member_birth`  
`= (select min(member_birth) from member);`



- 문제 2고객 아이디, 이름, 이용횟수를 이용횟수 내림차순으로 정렬하여 출력하시오  
 select m.member\_id, m.member\_name,count(\*) as 이용횟수  
 from member m join rent\_info r  
 on m.member\_id = r.member\_id  
 group by m.member\_id  
 order by 이용횟수 DESC;

The screenshot shows a SQL IDE interface. The top pane displays a query with the following text:

```

1  -- 고객 아이디, 이름, 이용횟수를 이용횟수 내림차순으로 정렬하여 출력하시오
2  • select m.member_id, m.member_name,count(*) as 이용횟수
3    from member m join rent_info r
4    on m.member_id = r.member_id
5    group by m.member_id
6    order by 이용횟수 DESC;
  
```

The bottom pane shows the 'Result Grid' with the following data:

| member_id  | member_name | 이용<br>횟수 |
|------------|-------------|----------|
| ivymartin  | 정수민         | 2        |
| miayoung   | 강서연         | 2        |
| rachel_w   | 김유진         | 2        |
| cara_green | 박지민         | 2        |

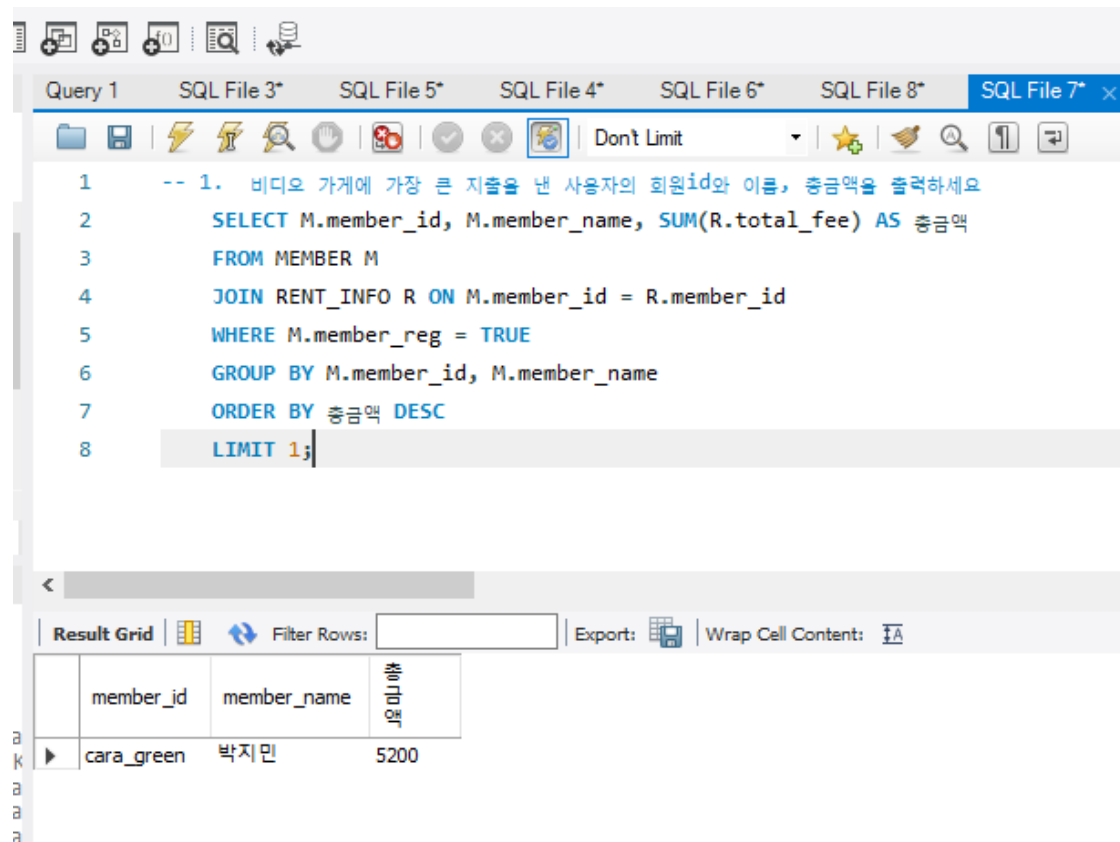
On the left side, a 'Schemas' panel shows a tree view with 'member' and 'rent\_info' tables. The 'member' table is selected, showing its columns: member\_id (PK), member\_name, member\_reg, member\_pwd, member\_phone, member\_birth, member\_add, and member\_reg.

다훈

```

-- 1. 비디오 가게에 가장 큰 지출을 낸 사용자의 회원id와 이름, 총금액을 출력하세요
SELECT M.member_id, M.member_name, SUM(R.total_fee) AS 총금액
FROM MEMBER M
JOIN RENT_INFO R ON M.member_id = R.member_id
WHERE M.member_reg = TRUE
GROUP BY M.member_id, M.member_name
ORDER BY 총금액 DESC
LIMIT 1;
  
```





- 2. 고객들이 가장 많이 대여한 비디오를 검색하여 그 비디오의 장르의 번호를 출력하세요  
SELECT video\_id, genre\_id  
FROM VIDEO  
WHERE video\_id = (  
SELECT video\_id  
FROM RENT\_INFO  
GROUP BY video\_id  
ORDER BY COUNT(video\_id) DESC  
LIMIT 1  
);

```

1  -- 2. 고객들이 가장 많이 대여한 비디오를 검색하여 그 비디오의 장르의 번호를 출력하세요
2  SELECT video_id, genre_id
3  FROM VIDEO
4  WHERE video_id = (
5      SELECT video_id
6      FROM RENT_INFO
7      GROUP BY video_id
8      ORDER BY COUNT(video_id) DESC
9      LIMIT 1
10 )

```

| video_id | genre_id |
|----------|----------|
| 45       | 50       |

- 3. 블랙리스트 컬럼을 member 테이블에 추가하세요  
ALTER TABLE MEMBER ADD COLUMN blacklist BOOLEAN DEFAULT FALSE;

```

1  -- 3. 블랙리스트 컬럼을 member 테이블에 추가하세요
2  ALTER TABLE MEMBER ADD COLUMN blacklist BOOLEAN DEFAULT FALSE;

```

| #   | Time     | Action                                                                     | Message                                                  |
|-----|----------|----------------------------------------------------------------------------|----------------------------------------------------------|
| 101 | 16:21:45 | update rent_info set actual_return_date = now() where actual_return_dat... | 0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0 |
| 102 | 16:22:53 | select member_id, member_name, member_birth from member where me...        | 2 row(s) returned                                        |
| 103 | 16:23:26 | select m.member_id, m.member_name, count(*) as 이용횟수 from memb...           | 46 row(s) returned                                       |
| 104 | 16:23:55 | SELECT M.member_id, M.member_name, SUM(R.total_fee) AS 총금액 ...             | 1 row(s) returned                                        |
| 105 | 16:24:25 | SELECT video_id, genre_id FROM VIDEO WHERE video_id = ( SEL...             | 1 row(s) returned                                        |
| 106 | 16:24:47 | ALTER TABLE MEMBER ADD COLUMN blacklist BOOLEAN DEFAULT...                 | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0   |

민성

1. 99년도 이후에 출시된 비디오를 빌린 회원의  
회원id,비디오이름,비디오출시일,대여날짜를 출력하세요  
(대여날짜는 내림차순으로 정렬)

```
select r.member_id, v.video_title,v.video_release_date,r.rent_date
from RENT_INFO r join VIDEO v
on r.video_id = v.video_id
where v.video_release_date > "99-12-31"
order by rent_date DESC;
```

The screenshot shows a SQL IDE window with a query editor and a result grid. The query editor contains the following SQL code:

```
1  -- 1. 99년도 이후에 출시된 비디오를 빌린 회원의
2  -- 회원id,비디오이름,비디오출시일,대여날짜를 출력하세요
3  -- (대여날짜는 내림차순으로 정렬)
4  • select r.member_id, v.video_title,v.video_release_date,r.rent_date
5  from RENT_INFO r join VIDEO v
6  on r.video_id = v.video_id
7  where v.video_release_date > "99-12-31"
8  order by rent_date DESC;
```

The result grid displays the following data:

| member_id  | video_title                           | video_release_date | rent_date  |
|------------|---------------------------------------|--------------------|------------|
| gracedavis | The Incredibles                       | 2004-11-05         | 2024-01-12 |
| miaw_123   | Gone Girl                             | 2014-10-03         | 2023-12-14 |
| queensam   | Anchorman: The Legend of Ron Burgundy | 2004-07-09         | 2023-11-21 |
| paul_clark | Superbad                              | 2007-08-17         | 2023-08-09 |
| paulgar    | Paranormal Activity                   | 2009-09-25         | 2023-07-15 |

- 2 -1 관람등급마다 비디오가 몇개 있는지를 구해서 관람등급,개수를 출력한다(group by함수를 사용할 것.)

```
select video_rating as 관람등급 ,count(*) as 개수 from VIDEO group by video_rating;
```

Query 1 SQL File 3\* SQL File 5\* SQL File 4\* SQL File 6\* SQL File 8\* SQL File 7\* x SQLAddit

1 -- 2 -1 관람등급마다 비디오가 몇개 있는지를 구해서 관람등급, 개수를 출력한다 (group by 함수를 사용할 것.)  
 2 • select video\_rating as 관람등급, count(\*) as 개수 from VIDEO group by video\_rating;  
 3  
 4  
 5 -- 2 -2 관람등급을 12세 이상, 15세 이상을 전체이용가로 변경한다  
 6 • update VIDEO set video\_rating='전체' where video\_rating='12세 이상' OR video\_rating='15세 이상';  
 7

Result Grid Filter Rows: Export: Wrap Cell Content: IA

| 관람등급   | 개수 |
|--------|----|
| 전체     | 32 |
| 12세 이상 | 17 |
| 15세 이상 | 37 |
| 19세 이상 | 14 |

Result 63 x Read Only Context

Output

2 -2 관람등급을 12세 이상, 15세 이상을 전체이용가로 변경한다

update VIDEO set video\_rating='전체' where video\_rating='12세 이상' OR video\_rating='15세 이상';

3  
 4  
 5 -- 2 -2 관람등급을 12세 이상, 15세 이상을 전체이용가로 변경한다  
 6 • update VIDEO set video\_rating='전체' where video\_rating='12세 이상' OR video\_rating='15세 이상';  
 7

Output

Action Output

| #     | Time     | Action                                                                        | Message                                                     |
|-------|----------|-------------------------------------------------------------------------------|-------------------------------------------------------------|
| ✓ 104 | 16:23:55 | SELECT M.member_id, M.member_name, SUM(R.total_fee) AS 총금액 ...                | 1 row(s) returned                                           |
| ✓ 105 | 16:24:25 | SELECT video_id, genre_id FROM VIDEO WHERE video_id = ( SEL...                | 1 row(s) returned                                           |
| ✓ 106 | 16:24:47 | ALTER TABLE MEMBER ADD COLUMN blacklist BOOLEAN DEFAULT...                    | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0      |
| ✓ 107 | 16:25:23 | select r.member_id, v.video_title, v.video_release_date, r.rent_date from ... | 35 row(s) returned                                          |
| ✓ 108 | 16:25:51 | select video_rating as 관람등급, count(*) as 개수 from VIDEO group by ...           | 4 row(s) returned                                           |
| ✓ 109 | 16:26:07 | update VIDEO set video_rating='전체' where video_rating='12세 이상' O...           | 54 row(s) affected Rows matched: 54 Changed: 54 Warnings: 0 |

## CRUD

member 테이블

회원가입하려한다

회원가입하기전에 아이디가 있는지 확인하기 위해 중복검사를한다.

```
select * from member where member_id="minsung7139";
```

중복이 없다는걸 확인하고 회원가입한다.

```
INSERT INTO MEMBER(member_id, member_name, member_pwd, member_phone,
member_birth, member_add, member_reg) VALUES
('minsung7139', '김민성', 'b394ec2f8e1e7088c184c45db05b29562d80e06d', '010-7139-
5699','99/08/28', '서울특별시 구로구 구로동 123', TRUE);
```

쓰던 아이디가 아니라 익숙하지 않아 아이디명을 변경하려고 한다.

```
update MEMBER set member_id = 'minsung5699' where member_phone='010-7139-
5699';
```

회원을 탈퇴하려고 한다.

```
delete from MEMBER where member_id = 'minsung5699';
```

GENRE테이블

```
INSERT INTO genre VALUES(110,"사극",2000,200,0);
```

```
select * from GENRE;
```

```
update genre set genre_name = "성인" where genre_id=110;
```

```
delete from GENRE where genre_id=110;
```

RENT\_INFO 테이블

```
INSERT INTO RENT_INFO VALUES(55,"minsung7139",39, NOW(), NOW()+INTERVAL 2
DAY,NULL,0,2000);
```

```
select * from RENT_INFO where rent_id = 55;
```

```
update RENT_INFO set rent_id=52 where rent_id=55;
```

```
delete from RENT_INFO where rent_id=52;
```

VIDEO 테이블

```
insert INTO VIDEO VALUES(101,"up",100,"전체","PIXAR","2009-07-30",1);
```

```
select * from VIDEO where video_id=101;
```

```
update VIDEO set video_title = "UP" where video_title="up";
```

```
delete from VIDEO where video_id = 101;
```