데이터 베이스
Term Project

02반 03조
201202156 오희빈
201202287 제갈수민

목차

1. 개요

1.1. 프로젝트 개요 및 배경

1.1.1. 개요

영화관에서 운영하는 영화예약 시스템에 대해서 데이터베이스를 이용해서 프로젝트를 만든다.

1.1.2. 배경

영화관 예약 시스템에 대해서 실제로 일어날 수 있는 일과 어떤 식의 데이터를 가져야 하는지 이해를 하고 각 데이터간의 관계와 구조를 통해 프로젝트 에 필요한 데이터베이스를 구현한다.

2. 영화 예매 시스템 시나리오 및 동작 과정

2.1. 고객 시나리오

2.1.1. 회원 정보 시나리오

2.1.1.1. 회원가입 동작과정

회원ID 중복 확인후 이름, 비밀번호, 전화번호, 주소, 생년월일 순으로 입력

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 2
What do you want to do? (1: log-in, 2: sign up, 3: exit) : 2
ID : heebin
You can use this ID.
Name : 오희빈
PW : 1234
Phone number : 01072373239
Address : 충북청주시분평동
Birth(e.g. 93/07/12) : 93/02/04
데이터 삽입 성공
```

2.1.1.2. 로그인 절차 및 동작과정

입력한 회원ID가 테이블에 존재하는지 확인 -〉 패스워드가 같은지 확인 -〉 로그인

(1) 로그인 성공

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 2
What do you want to do? (1: log-in, 2: sign up, 3: exit) : 1
What is your ID? : heebin
What is your pw? : 1234
Log-in success!
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 2
```

(2) 비밀번호 불일치시 로그인 실패

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 2
What do you want to do? (1: log-in, 2: sign up, 3: exit) : 1
What is your ID? : sumin
What is your pw? : 1234
Fail to Log-in
```

2.1.1.3. 회원정보수정 절차 및 동작과정

수정하고자하는 회원정보 선택 후 수정

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 3
sumin
=====================================================
This is your information
-----------------------------------------------------
ID : sumin
Name : 제갈수민
Password : abcd
Phone number : 01053787830
Adderss : 대전광역시중동
Birth : 1993-07-12 00:00:00
Point : 100
=====================================================
What do you want to change? (1: name, 2: password, 3: phone number, 4: address, 5: birth, 6: exit)3
How do you want to change? : 01024351234
데이터 수정 성공
What do you want to change? (1: name, 2: password, 3: phone number, 4: address, 5: birth, 6: exit)5
How do you want to change?(e.g. 93/07/12) : 91/12/23
데이터 수정 성공
What do you want to change? (1: name, 2: password, 3: phone number, 4: address, 5: birth, 6: exit)6
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) :
```

2.1.1.4. 회원탈퇴 절차 및 동작과정

로그인된 회원 정보 삭제

```
What do you want to do? (1: log-in, 2: sign up, 3: exit) : 1
What is your ID? : asd
What is your pw? : asd
Log-in success!
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 5
Are you sure to remove your account? (1: Yes, 2: No!) : 1
데이터 삭제 성공
What do you want to do? (1: log-in, 2: sign up, 3: exit) :
```

2.1.1.5. 로그아웃 절차 및 동작과정

로그아웃 선택

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 6
Complete log out
What do you want to do? (1: log-in, 2: sign up, 3: exit) : 3
Are you manager or customer? (1: manager, 2: customer 3: exit) :
```

2.1.2. 영화 검색 시나리오

2.1.2.1. 예매율에 따른 영화 검색 절차 및 동작과정

예매율이 높은 순으로 영화 제목을 보여준다.

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 2
================================================
          영화 제목          |      예매율
------------------------------------------------
             킹              |      40%
          라라랜드           |      30%
          판도라             |      25%
================================================
```

2.1.3. 예약 시나리오

2.1.3.1. 영화 예약을 위한 절차

영화관 이름들을 보여 준다 -〉 영화관 선택 -〉 영화관에서 상영 중인 영화를 보여 준다 -〉 영화 선택 -〉 선택한 영화의 상영일정을 보여준다. -〉 원하는 상영일정 선택 -〉 원하는 예매 매수 선택 -〉 남은 좌석에 대한 정보를 보여주고 좌석 선택 -〉 결제 시나리오(인터넷 결제, 현장 결제)

(1)인터넷 결제 선택

원하는 결제 방법 선택(인터넷 결제) -〉 (인터넷 결제 선택 시) 총 결제 금액과 현재 포인트를 보여준다 -〉 (포인트가 1000점 이상 있는 경우) 포인트 사용을 물어 본다 -〉 포인트 사용 금액을 물어본다 -〉 포인트가 차감되며 결제가 완료된다.

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 1
==================================================
영화관 이름 : CGV유성온천
영화관 이름 : 롯데시네마청주점
--------------------------------------------------
영화관 선택 : CGV유성온천
--------------------------------------------------
MOVIE_ID : a1    MOVIE_NAME : 판도라
MOVIE_ID : a2    MOVIE_NAME : 형
--------------------------------------------------
영화 선택 : 형
--------------------------------------------------
선택한 영화 : 형
상영번호 : 12      상영일정 : 2016-12-09 15:30:00
상영번호 : 13      상영일정 : 2016-12-09 18:00:00
--------------------------------------------------
상영번호 선택 : 13
예매할 매수 선택(남은 좌석 수 : 10) : 1
--------------------------------------------------
좌석선택<1~10>
<남은 예매 수(1)> : 5
데이터 삽입 성공
데이터 삽입 성공
데이터 수정 성공
데이터 수정 성공
결제하는 방법은? (1: 현장 결제, 2: 인터넷 결제) : 2
인터넷 결제를 시작합니다.
당신의 총 결제 금액은 10000입니다.
당신의 포인트는 0p 있습니다.
포인트 부족으로 결제에는 사용하실 수 없습니다.
데이터 삽입 성공
데이터 수정 성공
결제에 성공했습니다!
==================================================
```

(2)현장 결제 선택

원하는 결제 방법 선택(현장 결제) -〉 (관리자)현장 발권시나리오

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 1
==================================================
영화관 이름 : CGV유성온천
영화관 이름 : 롯데시네마청주점
--------------------------------------------------
영화관 선택 : CGV유성온천
--------------------------------------------------
MOVIE_ID : a1    MOVIE_NAME : 판도라
MOVIE_ID : a2    MOVIE_NAME : 형
--------------------------------------------------
영화 선택 : 형
--------------------------------------------------
선택한 영화 : 형
상영번호 : 12      상영일정 : 2016-12-09 15:30:00
상영번호 : 13      상영일정 : 2016-12-09 18:00:00
--------------------------------------------------
상영번호 선택 : 13
예매할 매수 선택(남은 좌석 수 : 9) : 2
--------------------------------------------------
좌석선택<1~10>
<예매된 좌석(5 ), 남은 예매 수(2)> : 3
데이터 삽입 성공
데이터 삽입 성공
데이터 수정 성공
데이터 수정 성공
좌석선택<1~10>
<예매된 좌석(3 5 ), 남은 예매 수(1)> : 8
데이터 삽입 성공
데이터 수정 성공
데이터 수정 성공
결제하는 방법은? (1: 현장 결제, 2: 인터넷 결제) : 1
==================================================
```

2.1.4. 예매 정보 확인 및 취소 시나리오

2.1.4.1. 예매 정보 와 티켓 확인

예매 정보 확인 선택 -〉 티켓 확인 선택 -〉 예매 번호 선택 -〉 티켓 정보 출력

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 4
==========================================================================================
   예약번호          영화 제목       영화관          상영 시간       상영관
------------------------------------------------------------------------------------------
       29                    형     CGV유성온천    2016-12-09 18:00:00          1
       30                    형     CGV유성온천    2016-12-09 18:00:00          1
==========================================================================================

What do you want to do? (1: show tickets of reservation, 2: delete reservation, 3: exit) :
1
What reservation do you want to see more? (Input the reservation number) :
30
==========================================================================================
   티켓번호       좌석번호       예약번호         영화관      상영관
------------------------------------------------------------------------------------------
       44             3            30          CGV유성온천       1
       45             8            30          CGV유성온천       1
==========================================================================================
```

2.1.4.2. 예매 취소

예매 취소할 예매 번호 선택 -〉 예매 취소 -〉 포인트 차감 -〉 티켓팅 수 차감

(1)동작과정

```
What do you want to do? (1: booking a movie, 2: search movies, 3: modify your information, 4: check reservation, 5: secession, 6:log-out) : 4
=============================================================================================
   예약번호          영화 제목          영화관          상영 시간       상영관
---------------------------------------------------------------------------------------------
       27                    판도라      CGV유성온천     2016-12-19 20:00:00          2
       28                   라라랜드     롯데시네마청주점     2016-12-12 15:00:00          1
=============================================================================================

What do you want to do? (1: show tickets of reservation, 2: delete reservation, 3: exit) :
2
What reservation do you want to delete? (Input the reservation number) :
27
데이터 삭제 성공
데이터 수정 성공
데이터 수정 성공
데이터 수정 성공
What do you want to do? (1: show tickets of reservation, 2: delete reservation, 3: exit) :
3
```

(2) 예매취소 전과 후 포인트 차감

```
heebin
============================================================
This is your information
------------------------------------------------------------
ID : heebin
Name : 오희빈
Password : 1234
Phone number : 01072373239
Adderss : 충북청주시분평동
Birth : 1993-02-04 00:00:00
Point : 500
============================================================
```

```
heebin
============================================================
This is your information
------------------------------------------------------------
ID : heebin
Name : 오희빈
Password : 1234
Phone number : 01072373239
Adderss : 충북청주시분평동
Birth : 1993-02-04 00:00:00
Point : 300
============================================================
```

(3) 예매 취소 전과 후 티켓팅 수

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 1
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 5
================================================
            고객 ID      |    티켓예매횟수
------------------------------------------------
                heebin      |        5
                 sumin      |        3
================================================

What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 5
================================================
            고객 ID      |    티켓예매횟수
------------------------------------------------
                 sumin      |        3
                heebin      |        3
================================================
```

2.2. 관리자 시나리오

2.2.1. 영화 정보 관리

2.2.1.1. 영화 정보 등록

영화 ID, 영화 제목, 감독, 출연, 등급, 주요 정보, 영화제목, 예매율, 러닝타임 입력

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 1
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 1
What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3: 영화 정보 수정, 4: exit) : 1
==============================================
영화ID? : a1
영화제목? : 판도라
RuningTime? : 136
영화정보? : 최대 규모의 장전에 이어 원자력 폭발 사고까지 일어나는 스릴러 영화
GRADE?(전체, 12, 15, 19) : 12
감독? : 박정우
배우? : 김남길,김영애,윤정희
==============================================
데이터 삽입 성공
```

2.2.1.2. 영화 정보 삭제

영화 정보 출력 -〉 삭제할 영화 ID 선택 -〉 삭제 완료

```
What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3: 영화 정보 수정, 4: exit) : 2
==================================================
ID : a2 MOVIE_NAME : 형
ID : asd MOVIE_NAME : 신비한 동물 사전
ID : a1 MOVIE_NAME : 판도라
==================================================
삭제할 영화ID? : asd
데이터 삭제 성공
delete success!
What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3: 영화 정보 수정, 4: exit) :
```

2.2.1.3. 영화 정보 수정

영화 정보 출력 -〉 수정할 영화 ID 선택 -〉 수정할 정보 선택 -〉 수정 -〉 수정 완료

```
What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3: 영화 정보 수정, 4: exit) : 3
==============================================
ID : a2 MOVIE_NAME : 형
ID : a1 MOVIE_NAME : 판도라
==============================================
수정할 영화ID? : a2
무엇을 수정하시겠습니까? (1: MOVIE_NAME, 2: RESERVATION_RATE, 3: RUNINGTIME, 4: INFORMATION5: GRADE, 6: DIRECTOR, 7: MOVIE_CAST, 8: exit) : 2
수정할 RESERVATION_RATE?(100이하) : 40
데이터 수정 성공
무엇을 수정하시겠습니까? (1: MOVIE_NAME, 2: RESERVATION_RATE, 3: RUNINGTIME, 4: INFORMATION5: GRADE, 6: DIRECTOR, 7: MOVIE_CAST, 8: exit) : 5
수정할 GRADE?(전체, 12, 15, 19) : 전체
데이터 수정 성공
무엇을 수정하시겠습니까? (1: MOVIE_NAME, 2: RESERVATION_RATE, 3: RUNINGTIME, 4: INFORMATION5: GRADE, 6: DIRECTOR, 7: MOVIE_CAST, 8: exit) : 3
수정할 RUNINGTIME? : 112
데이터 수정 성공
무엇을 수정하시겠습니까? (1: MOVIE_NAME, 2: RESERVATION_RATE, 3: RUNINGTIME, 4: INFORMATION5: GRADE, 6: DIRECTOR, 7: MOVIE_CAST, 8: exit) : 8
What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3: 영화 정보 수정, 4: exit) :
```

2.2.2. 영화관 정보 관리 시나리오

2.2.2.1. 영화관 정보 등록

영화관 이름, 영화관 주소, 총 좌석 수, 영화관 전화번호, 상영관 수 순으로 입력

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 1
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 2
What do you want to do? (1: Upload cinema, 2: delete cinema, 3: modify cinema information, 4: exit) : 1
================================================
영화관이름? : CGV유성온천
영화관주소? : 대전광역시궁동
영화관전화번호? : 01023124563
영화관총좌석수? : 15
영화관 전체 상영관수? : 2
데이터 삽입 성공
================================================
```

2.2.2.2. 영화관 정보 수정

영화관 정보 출력 -〉 수정할 영화관 이름 선택 -〉 수정할 정보 선택 -〉 수정-〉 수정 완료

```
What do you want to do? (1: Upload cinema, 2: delete cinema, 3: modify cinema information, 4: exit) : 3
================================================
CINEMA_NAME : CGV유성온천
CINEMA_NAME : 롯데시네마청주점
수정할 영화관이름? : 롯데시네마청주점
무엇을 수정하시겠습니까? (1: CINEMA_ADDRESS, 2: CINEMA_PHONENUMBER , 3: CINEMA_TOTAL_SEAT_NUMBER 4: TOTAL_SCREEN_NUMBER, 5: exit) : 3
수정할 CINEMA_TOTAL_SEAT_NUMBER? : 14
데이터 수정 성공
무엇을 수정하시겠습니까? (1: CINEMA_ADDRESS, 2: CINEMA_PHONENUMBER , 3: CINEMA_TOTAL_SEAT_NUMBER 4: TOTAL_SCREEN_NUMBER, 5: exit) : 1
수정할 CINEMA_ADDRESS? : 청주시성안길
데이터 수정 성공
무엇을 수정하시겠습니까? (1: CINEMA_ADDRESS, 2: CINEMA_PHONENUMBER , 3: CINEMA_TOTAL_SEAT_NUMBER 4: TOTAL_SCREEN_NUMBER, 5: exit) : 2
수정할 CINEMA_PHONENUMBER? : 0432351234
데이터 수정 성공
```

2.2.2.3. 영화관 정보 삭제

영화관 정보 출력 -〉 삭제할 영화관 이름 선택 -〉 삭제 완료

```
What do you want to do? (1: Upload cinema, 2: delete cinema, 3: modify cinema information, 4: exit) : 2
================================================
CinemaName : CGV유성온천
CinemaName : 롯데시네마
삭제할 영화관이름? : 롯데시네마
데이터 삭제 성공
delete success!
```

2.2.3. 상영관 정보 관리 시나리오

2.2.3.1. 상영관 정보 등록

영화관 정보 출력 -〉 영화관 선택 -〉 영화관의 상영관 수와 총 좌석 수를 출력 -〉 만들어진 상영관을 출력 -〉 만들 상영관 번호 입력 -〉 남은 영화관 총 좌석 수를 보여주고 상영관의 좌석 수를 선택 -〉 등록 완료

```
================================================
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 3
What do you want to do? (1: Upload screen, 2: exit) : 1
================================================
CinemaName : CGV유성온천
CinemaName : 롯데시네마청주점
================================================
상영관을 추가할 영화관 선택 : 롯데시네마청주점
================================================
Total Screen Number : 3
Left Total Seat Number : 14
================================================
<만들어진 상영관 : >
상영관 번호(Number) : 1
원하는 좌석 수 <남은 전체 좌석수(14)> : 6
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
<만들어진 상영관 : 1 >
상영관 번호(Number) : 2
원하는 좌석 수 <남은 전체 좌석수(8)> : 4
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
데이터 삽입 성공
<만들어진 상영관 : 1 2 >
상영관 번호(Number) : 3
원하는 좌석 수 <남은 전체 좌석수(4)> : 4
데이터 삽입 성공
```

2.2.4. 상영 일정 정보 관리 시나리오

2.2.4.1. 상영 일정 정보 등록

존재하는 영화관 출력 -〉 상영을 추가할 영화관 선택 -〉 영화관의 상영관 출력 -〉 원하는 상영관 선택 -〉 선택한 상영관의 상영일정과 영화 제목 출력-〉 상영 일정 입력 -〉 영화 정보 출력 -〉 상영할 영화 선택 -〉 상영 정보 등록 완료

(1)상영 일정이 없는 경우

```
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 4
What do you want to do? (1: 상영 정보 등록, 2: 상영 정보 삭제, 3: exit) : 1
==================================================
cinemaName : CGV유성온천
cinemaName : 롯데시네마청주점
==================================================
상영을 추가할 영화관 선택 : CGV유성온천
==================================================
screenNumber : 1
screenNumber : 2
==================================================
상영할 상영관 선택 : 1
==================================================
해당 상영관에는 상영일정이 없습니다.
==================================================
상영 일정 입력 (e.g 2013/12/12 15:20:00) : 2016/12/09 15:30:00
==================================================
movieName : 형   movieID : a2
movieName : 판도라        movieID : a1
movieName : 라라랜드      movieID : a3
==================================================
상영할 영화ID 선택 : a2
데이터 삽입 성공
```

(2)상영 일정이 있는 경우

```
What do you want to do? (1: 상영 정보 등록, 2: 상영 정보 삭제, 3: exit) : 1
==================================================
cinemaName : CGV유성온천
cinemaName : 롯데시네마청주점
==================================================
상영을 추가할 영화관 선택 : CGV유성온천
==================================================
screenNumber : 1
screenNumber : 2
==================================================
상영할 상영관 선택 : 1
==================================================
movieName : 형           screeningTime : 2016-12-09 15:30:00
==================================================
상영 일정 입력 (e.g 2013/12/12 15:20:00) : 2016/12/09 18:00:00
==================================================
movieName : 형   movieID : a2
movieName : 판도라        movieID : a1
movieName : 라라랜드      movieID : a3
==================================================
상영할 영화ID 선택 : a2
데이터 삽입 성공
```

2.2.4.2. 상영 일정 정보 삭제

영화관 정보 출력 -〉 상영일정을 삭제할 영화관 이름 선택 -〉 영화관의 상영 번호와 상영관 번호, 상영 일정 출력 -〉 삭제할 상영 번호 선택 -〉 삭제 완료

```
What do you want to do? (1: 상영 정보 등록, 2: 상영 정보 삭제, 3: exit) : 2
==================================================
cinemaName : CGV유성온천
cinemaName : 롯데시네마청주점
==================================================
상영 일정을 삭제할 영화관 선택 : CGV유성온천
==================================================
screeningNumber : 12    screenNumber : 1관      screeningTime : 2016-12-09 15:30:00
screeningNumber : 13    screenNumber : 1관      screeningTime : 2016-12-09 18:00:00
screeningNumber : 14    screenNumber : 2관      screeningTime : 2016-12-19 20:00:00
screeningNumber : 15    screenNumber : 2관      screeningTime : 2011-12-12 15:00:00
==================================================
상영 일정을 삭제할 상영 번호 선택 : 15
데이터 삭제 성공
delete success!
What do you want to do? (1: 상영 정보 등록, 2: 상영 정보 삭제, 3: exit) :
```

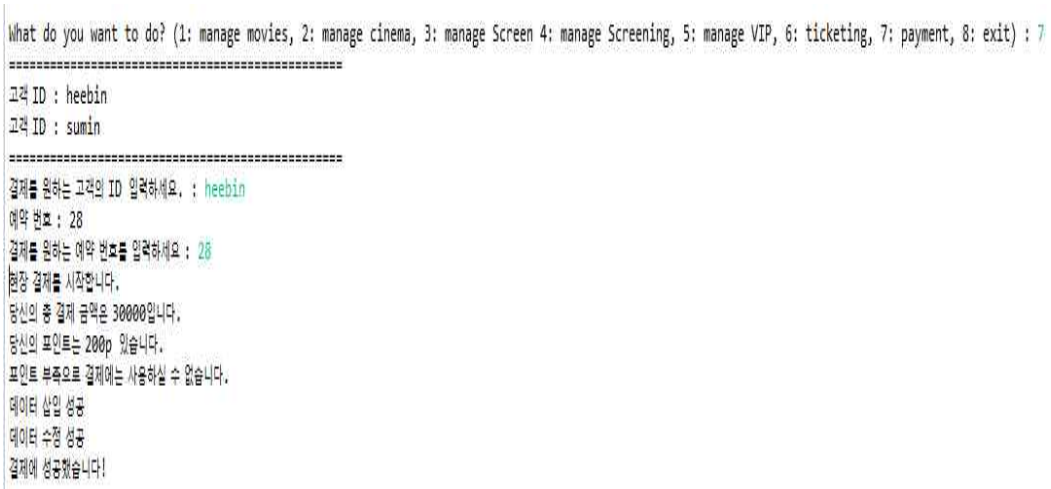
2.2.5. VIP 고객 관리 시나리오

고객의 영화 티켓 예매 수가 높은 순서로 VIP고객을 보여준다.

```
Are you manager or customer? (1: manager, 2: customer 3: exit) : 1
What do you want to do? (1: manage movies, 2: manage cinema, 3: manage Screen 4: manage Screening, 5: manage VIP, 6: ticketing, 7: payment, 8: exit) : 5
==================================================
          고객 ID      |    티켓예매횟수
--------------------------------------------------
                heebin      |        5
                 sumin      |        3
==================================================
```

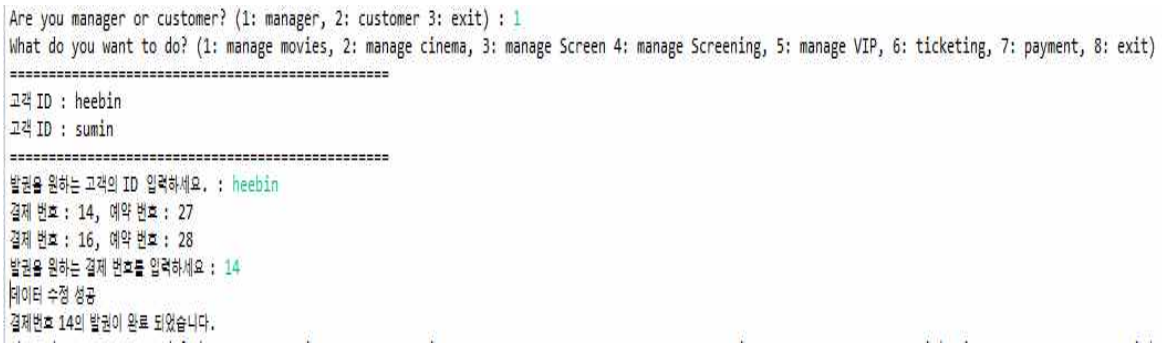
2.2.6. 현장 결제 시나리오

고객 ID 출력 -〉 결제를 원하는 고객 ID 입력 -〉 현장 결제를 선택한 예매 번호 출력 -〉 원하는 예매 번호 입력 -〉 총 금액과 현재 포인트 출력 -〉 포 인트가 1000점 이상인 경우 포인트 사용 유무를 확인 -〉 포인트 사용 시 포인트를 차감 -〉 결제 완료



2.2.7. 발권 시나리오

고객의 ID 출력 -〉 발권을 원하는 고객 ID 입력 -〉 입력한 고객 ID의 결제 내역 출력 -〉 발권을 원하는 결제 번호 입력 -〉 발권 완료



3. 설계 시 제약 조건

3.1. 고객

고객 ID는 고유한 값으로 null값이 들어가거나 중복될 수 없다.

3.2. 영화

등급은 (전체, 12, 15, 청소년 관람불가)중에 선택한다.

러닝타임은 분 단위로 입력한다.

3.3. 상영관

상영관 번호는 해당 영화관 상영관 수를 넘지 않는다.

상영관 좌석 수는 영화관 총 좌석 수를 넘지 않는다.

3.4. 좌석

좌석 번호는 해당 상영관 좌석 수를 넘지 않는다.

3.5. 예매

예매 좌석 선택은 해당 상영관 좌석 수를 넘지 않는다.

이미 예매된 좌석은 선택이 불가능 하다.

1번 예매 시 1장 이상의 티켓 예매가 가능하다.

3.6. 결제

결제 시 현장 결제와 인터넷 결제 중 하나를 선택한다.

사용 포인트는 고객의 포인트가 1000점 이상 일 때만 사용가능하다.

티켓 한 장당 100점의 적립 포인트를 부여한다.

3.7. 발권

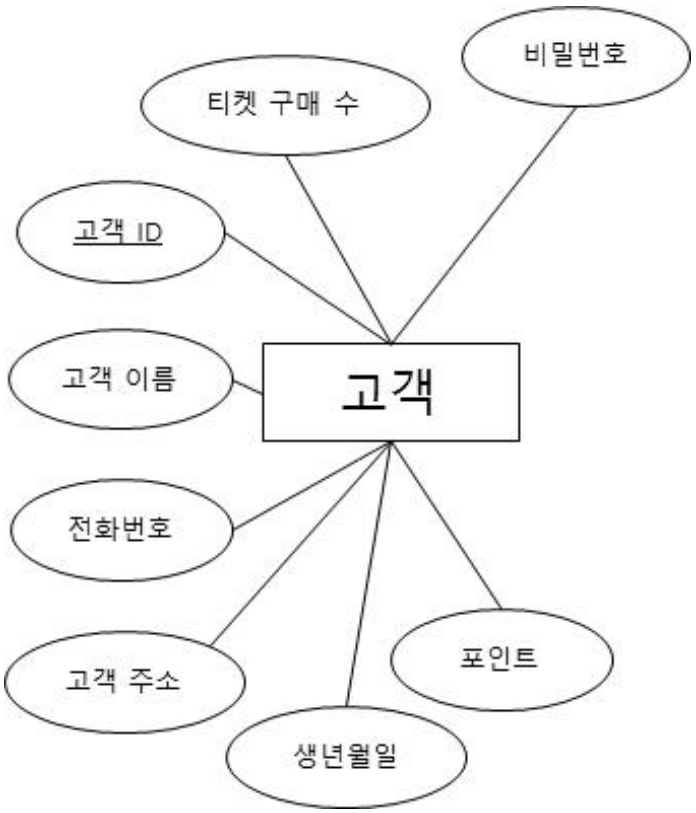
결제가 완료된 예매에 대해서만 발권이 가능하다.

3.8. 상영

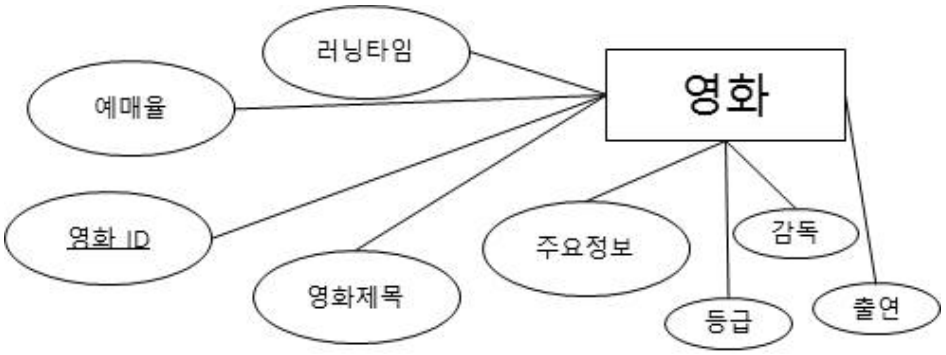
상영관에 등록된 상영 일정이 다른 시간이여야만 등록이 가능하다.

4. 영화 예매 시스템 ER-다이어그램

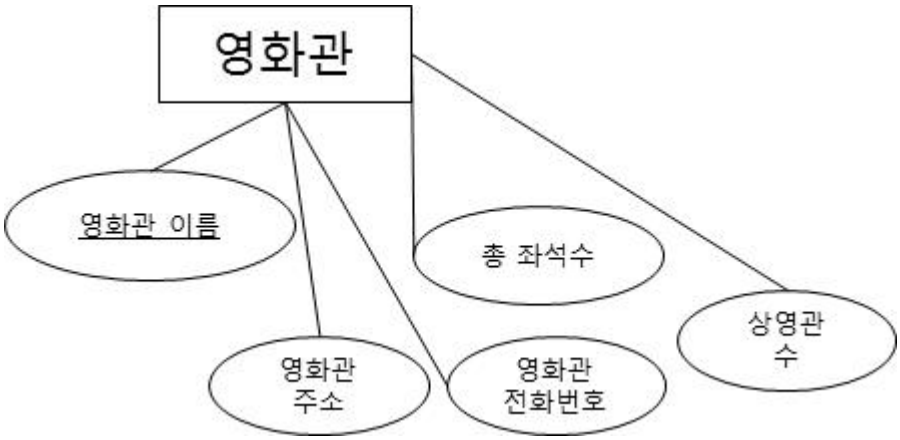
4.1. 고객



4.2. 영화

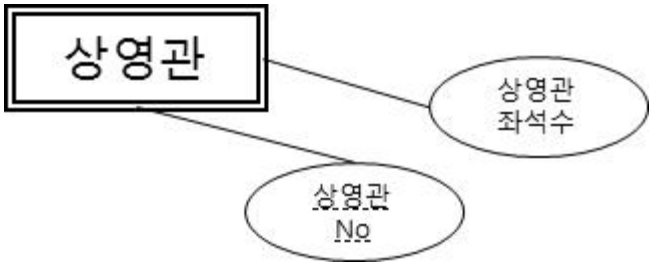


4.3. 영화관

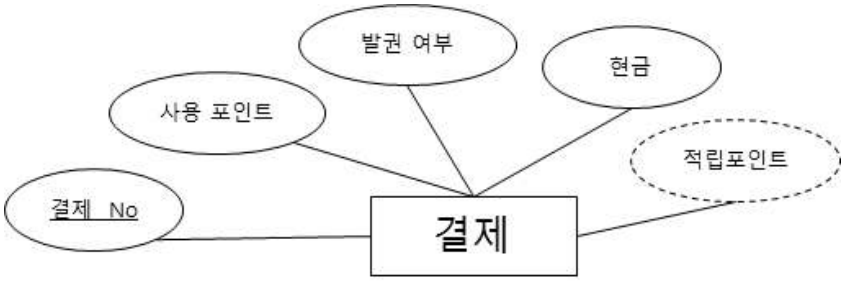


4.4. 티켓

4.5. 상영관

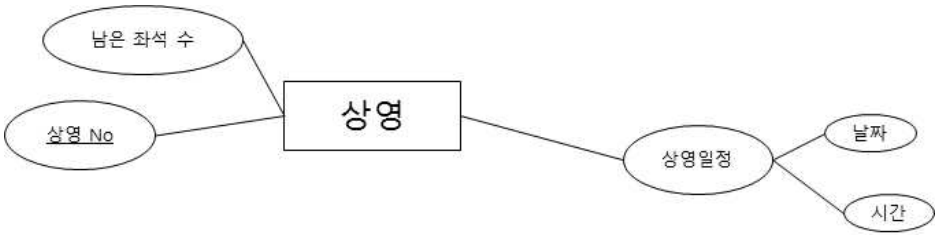


4.6. 결제



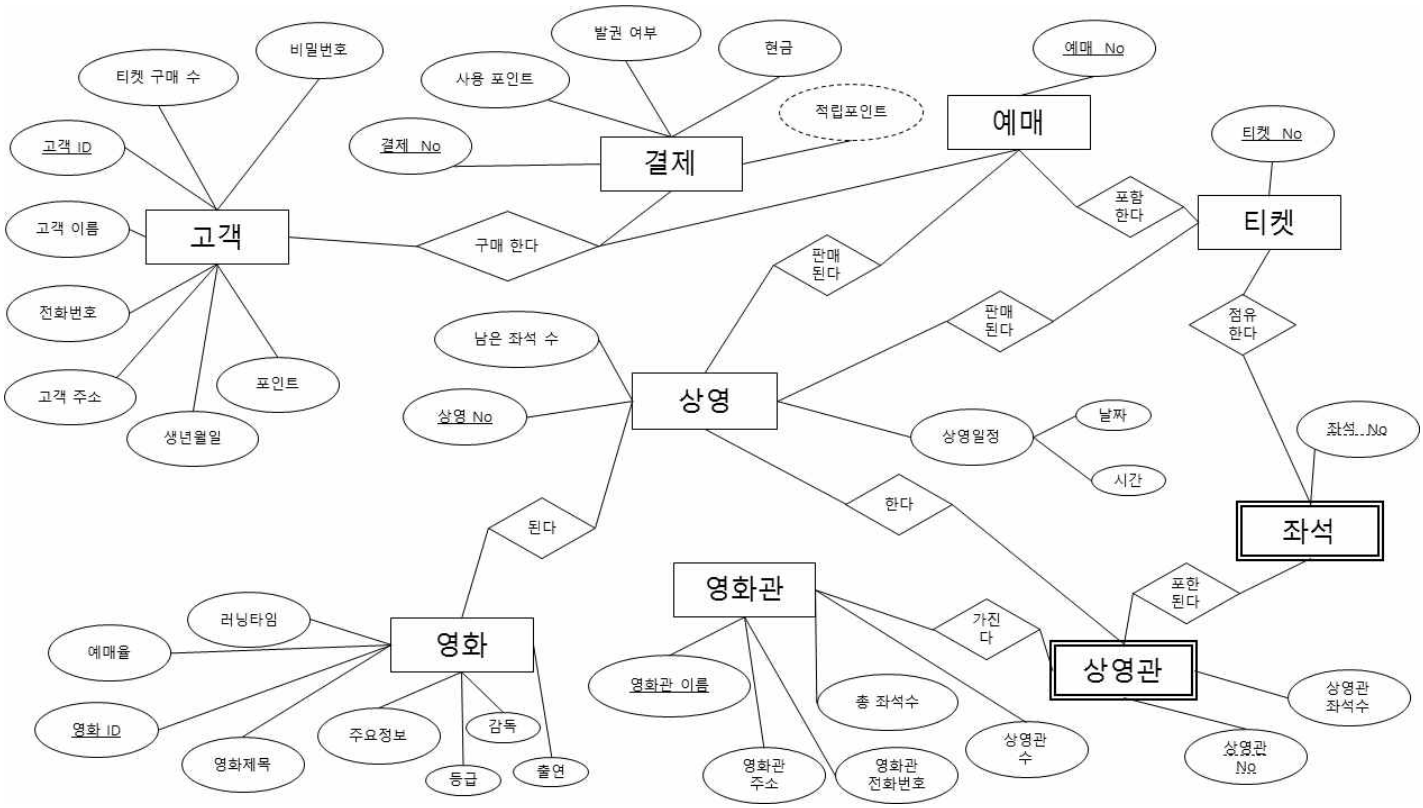
4.7. 예매

4.8. 상영

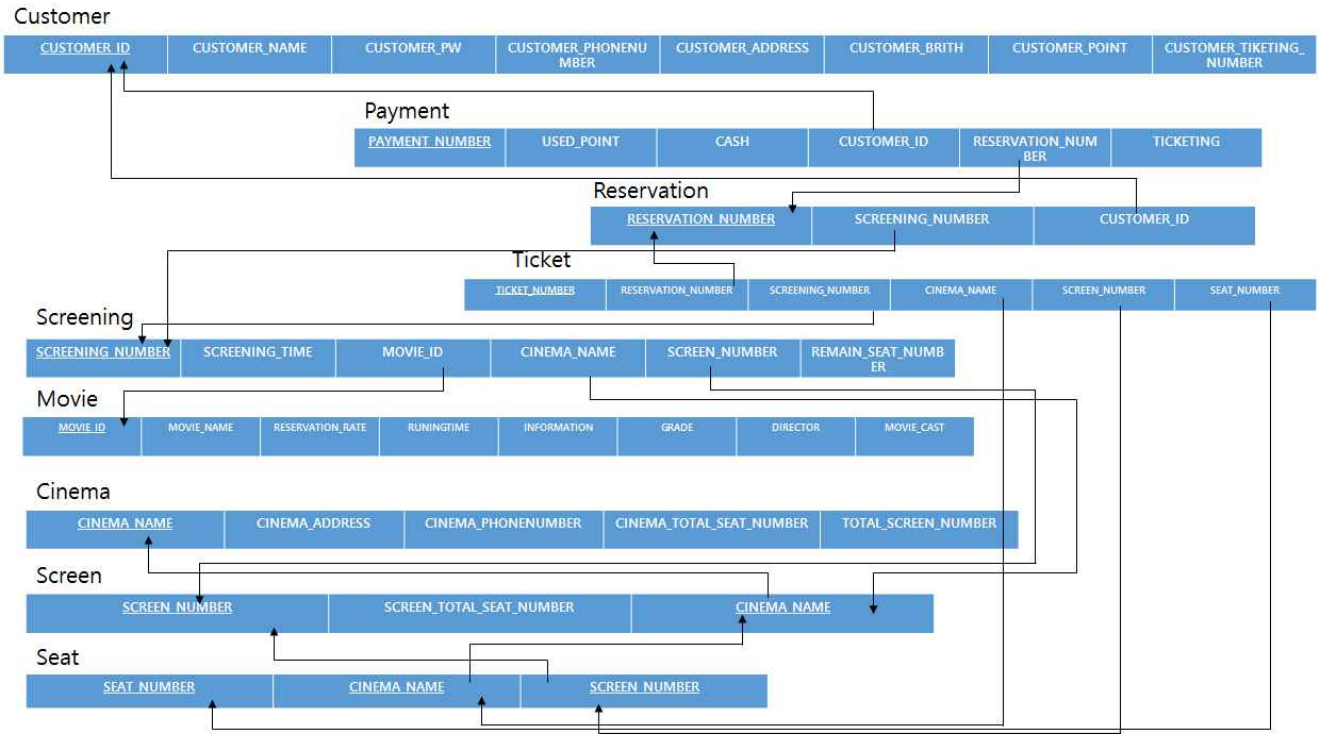


4.9. 좌석

4.10. 전체 ER-다이어그램


비밀번호
티켓 구매 수
고객 ID
고객 이름
고객
전화번호
고객 주소
생년월일
포인트
발권 여부
현금
사용 포인트
결제 No
결제
적립포인트
구매 한다
예매 No
예매
판매 된다
포함 한다
티켓 No
티켓
판매 된다
점유 한다
남은 좌석 수
상영 No
상영
상영일정
날짜
시간
한다
좌석 No
좌석
된다
포한 된다
러닝타임
예매율
영화
영화 ID
영화제목
주요정보
감독
등급
출연
영화관
가진 다
영화관 이름
총 좌석수
영화관 주소
영화관 전화번호
상영관 수
상영관
상영관 No
상영관 좌석수

5. 영화 예매 시스템 스키마 다이어그램



6. SQL DDL

6.1. Customer

```
--고객--
CREATE TABLE CUSTOMER (
  CUSTOMER_ID VARCHAR(10) NOT NULL,--고객ID--
  CUSTOMER_NAME VARCHAR(100) NOT NULL,--고객이름--
  CUSTOMER_PW VARCHAR(100) NOT NULL,--고객비밀번호--
  CUSTOMER_PHONENUMBER VARCHAR(100) NOT NULL,--고객전화번호--
  CUSTOMER_ADDRESS VARCHAR(100),--고객주소--
  CUSTOMER_BRITH DATE NOT NULL,--고객생년월일--
  CUSTOMER_POINT NUMBER DEFAULT 0,--고객의포인트--
  CUSTOMER_TIKETING_NUMBER NUMBER DEFAULT 0,--고객의티켓팅수--
  PRIMARY KEY (CUSTOMER_ID)
);
```

6.2. Payment

```
--결제--
CREATE TABLE PAYMENT (
  PAYMENT_NUMBER NUMBER NOT NULL,--결제번호--
  USED_POINT NUMBER,--사용된포인트
  CASH NUMBER,--현금--
  CUSTOMER_ID VARCHAR(10) NOT NULL,--고객ID--
  RESERVATION_NUMBER NUMBER NOT NULL,--예매번호--
  TICKETING NUMBER,--예매한티켓수--
  PRIMARY KEY (PAYMENT_NUMBER),
  FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID)
  ON DELETE CASCADE,
  FOREIGN KEY (RESERVATION_NUMBER) REFERENCES RESERVATION (RESERVATION_NUMBER)
  ON DELETE CASCADE
);
```

6.3. Reservation

```
--예매--
CREATE TABLE RESERVATION (
  RESERVATION_NUMBER NUMBER NOT NULL,--예매번호--
  SCREENING_NUMBER VARCHAR(10) NOT NULL,--상영번호--
  CUSTOMER_ID VARCHAR(10) NOT NULL,--고객ID--
  PRIMARY KEY (RESERVATION_NUMBER),
  FOREIGN KEY (SCREENING_NUMBER) REFERENCES SCREENING (SCREENING_NUMBER) ON DELETE CASCADE,
  FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID)
  ON DELETE CASCADE
);
```

6.4. Ticket

```
--티켓
CREATE TABLE TICKET (
  TICKET_NUMBER NUMBER NOT NULL,--티켓번호--
  SEAT_NUMBER VARCHAR(20) NOT NULL,--좌석번호--
  RESERVATION_NUMBER NUMBER NOT NULL,--예매번호--
  CINEMA_NAME VARCHAR(100) NOT NULL,--영화관이름--
  SCREEN_NUMBER NUMBER NOT NULL,--상영관번호--
  SCREENING_NUMBER VARCHAR(10) NOT NULL,--상영번호--
  PRIMARY KEY (TICKET_NUMBER),
  FOREIGN KEY (SEAT_NUMBER,SCREEN_NUMBER,CINEMA_NAME) REFERENCES SEAT (SEAT_NUMBER,SCREEN_NUMBER,CINEMA_NAME)
  ON DELETE CASCADE,
  FOREIGN KEY (RESERVATION_NUMBER) REFERENCES RESERVATION (RESERVATION_NUMBER)
  ON DELETE CASCADE,
  FOREIGN KEY (SCREENING_NUMBER ) REFERENCES SCREENING (SCREENING_NUMBER )
  ON DELETE CASCADE
);
```

6.5. Screening

```
--상영--
CREATE TABLE SCREENING (
  SCREENING_NUMBER VARCHAR(10) NOT NULL,--상영번호--
  SCREENING_TIME DATE NOT NULL,--상영일정--
  MOVIE_ID VARCHAR(10) NOT NULL,--영화ID--
  CINEMA_NAME VARCHAR(100) NOT NULL,--영화관이름--
  SCREEN_NUMBER NUMBER NOT NULL,--상영번호--
  REMAIN_SEAT_NUMBER NUMBER DEFAULT 0,--남은좌석수--
  PRIMARY KEY (SCREENING_NUMBER),
  FOREIGN KEY (MOVIE_ID) REFERENCES MOVIE (MOVIE_ID)
  ON DELETE CASCADE,
  FOREIGN KEY (SCREEN_NUMBER,CINEMA_NAME) REFERENCES SCREEN (SCREEN_NUMBER,CINEMA_NAME)
  ON DELETE CASCADE
);
```

6.6. Movie

```
--영화--
CREATE TABLE MOVIE (
  MOVIE_ID VARCHAR(10) NOT NULL,--영화ID--
  MOVIE_NAME VARCHAR(100) NOT NULL,--영화이름--
  RESERVATION_RATE NUMBER CHECK (RESERVATION_RATE <= 100),--영화예매율--
  RUNINGTIME NUMBER,--상영시간--
  INFORMATION VARCHAR(1000),--영화정보--
  GRADE VARCHAR(100) NOT NULL CHECK (GRADE IN ('전체', '12', '15', '19')),--등급--
  DIRECTOR VARCHAR(100) NOT NULL,--감독--
  MOVIE_CAST VARCHAR(100) NOT NULL,--출연--
  PRIMARY KEY (MOVIE_ID)
);
```

6.7. Cinema

```
--영화관--
CREATE TABLE CINEMA (
  CINEMA_NAME VARCHAR(100) NOT NULL,--영화관이름--
  CINEMA_ADDRESS VARCHAR(100),--영화관주소--
  CINEMA_PHONENUMBER VARCHAR(100) NOT NULL,--영화관전화번호--
  CINEMA_TOTAL_SEAT_NUMBER NUMBER,--전체좌석수--
  TOTAL_SCREEN_NUMBER NUMBER,--전체상영관수--
  PRIMARY KEY (CINEMA_NAME)
);
```

6.8. Screen

```
--상영관--
CREATE TABLE SCREEN (
  SCREEN_NUMBER NUMBER NOT NULL,--상영관번호--
  SCREEN_TOTAL_SEAT_NUMBER NUMBER,--상영관좌석수--
  CINEMA_NAME VARCHAR(100) NOT NULL,--영화관이름--
  PRIMARY KEY (SCREEN_NUMBER, CINEMA_NAME),
  FOREIGN KEY (CINEMA_NAME) REFERENCES CINEMA (CINEMA_NAME)
  ON DELETE CASCADE
);
```

6.9. Seat

```
--좌석--
CREATE TABLE SEAT (
  SEAT_NUMBER VARCHAR(50) NOT NULL,--좌석번호--
  CINEMA_NAME VARCHAR(100) NOT NULL,--영화관이름--
  SCREEN_NUMBER NUMBER NOT NULL,--상영관번호--
  PRIMARY KEY (SEAT_NUMBER, CINEMA_NAME, SCREEN_NUMBER),
  FOREIGN KEY (SCREEN_NUMBER,CINEMA_NAME) REFERENCES SCREEN (SCREEN_NUMBER,CINEMA_NAME)
  ON DELETE CASCADE
);
```

7. SQL DML

7.1. Trigger (on update cascade)

```
CREATE OR REPLACE TRIGGER TRI_CINEMA
AFTER UPDATE ON CINEMA
FOR EACH ROW
BEGIN
 UPDATE SCREEN
 SET CINEMA_NAME =:NEW.CINEMA_NAME
 WHERE CINEMA_NAME =:OLD.CINEMA_NAME ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_MOVIE
AFTER UPDATE ON MOVIE
FOR EACH ROW
BEGIN
 UPDATE SCREENING
 SET MOVIE_ID =:NEW.MOVIE_ID
 WHERE MOVIE_ID =:OLD.MOVIE_ID;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_RESERVATION
AFTER UPDATE ON RESERVATION
FOR EACH ROW
BEGIN
 UPDATE TICKET
 SET RESERVATION_NUMBER =:NEW.RESERVATION_NUMBER
 WHERE RESERVATION_NUMBER =:OLD.RESERVATION_NUMBER ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREENING_REMAIN
AFTER DELETE ON RESERVATION
FOR EACH ROW
BEGIN
 UPDATE SCREENING
 SET SCREENING_REMAIN_SEAT_NUMBER =:NEW.SCREENING_NUMBER
 WHERE SCREENING_NUMBER =:OLD.SCREENING_NUMBER ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREENING
AFTER UPDATE ON SCREENING
FOR EACH ROW
BEGIN
 UPDATE RESERVATION
 SET SCREENING_NUMBER =:NEW.SCREENING_NUMBER
 WHERE SCREENING_NUMBER =:OLD.SCREENING_NUMBER ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREEN
AFTER UPDATE ON SCREEN
FOR EACH ROW
BEGIN
 UPDATE SCREENING
 SET SCREEN_NUMBER =:NEW.SCREEN_NUMBER
 WHERE SCREEN_NUMBER =:OLD.SCREEN_NUMBER;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREEN2
AFTER UPDATE ON SCREEN
FOR EACH ROW
BEGIN
 UPDATE SCREENING
 SET CINEMA_NAME =:NEW.CINEMA_NAME
 WHERE CINEMA_NAME =:OLD.CINEMA_NAME;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREEN3
AFTER UPDATE ON SCREEN
FOR EACH ROW
BEGIN
 UPDATE SEAT
 SET SCREEN_NUMBER =:NEW.SCREEN_NUMBER
 WHERE SCREEN_NUMBER =:OLD.SCREEN_NUMBER;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SCREEN4
AFTER UPDATE ON SCREEN
FOR EACH ROW
BEGIN
 UPDATE SEAT
 SET CINEMA_NAME =:NEW.CINEMA_NAME
 WHERE CINEMA_NAME =:OLD.CINEMA_NAME ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SEAT1
AFTER UPDATE ON SEAT
FOR EACH ROW
BEGIN
 UPDATE TICKET
 SET SEAT_NUMBER =:NEW.SEAT_NUMBER
 WHERE SEAT_NUMBER  =:OLD.SEAT_NUMBER ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SEAT2
AFTER UPDATE ON SEAT
FOR EACH ROW
BEGIN
 UPDATE TICKET
 SET SCREEN_NUMBER =:NEW.SCREEN_NUMBER
 WHERE SCREEN_NUMBER =:OLD.SCREEN_NUMBER ;
END;
```

```
CREATE OR REPLACE TRIGGER TRI_SEAT3
AFTER UPDATE ON SEAT
FOR EACH ROW
BEGIN
 UPDATE TICKET
 SET CINEMA_NAME =:NEW.CINEMA_NAME
 WHERE CINEMA_NAME =:OLD.CINEMA_NAME ;
END;
```

7.2. Sequence

```
--자동으로 증가시키기위해 만든 시퀀스들--
CREATE SEQUENCE SNUM START WITH 1 INCREMENT BY 1 MAXVALUE 1000 CYCLE NOCACHE;--상영번호를 증가시키기위해만든 시퀀스--
CREATE SEQUENCE RESERVATENUM START WITH 1 INCREMENT BY 1 MAXVALUE 1000 CYCLE NOCACHE;--예약번호를 증가시키기위해만든 시퀀스--
CREATE SEQUENCE TICKETNUM START WITH 1 INCREMENT BY 1 MAXVALUE 1000 CYCLE NOCACHE;--티켓번호를 증가시키기위해만든 시퀀스--
CREATE SEQUENCE PAYMENTNUM START WITH 1 INCREMENT BY 1 MAXVALUE 1000 CYCLE NOCACHE;--결제번호를 증가시키기위해만든 시퀀스--
```

7.3. DML을 이용한 테이블 삽입 결과 화면

7.3.1. Customer

	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_PW	CUSTOMER_PHONENUMBER	CUSTOMER_ADDRESS	CUSTOMER_BRITH	CUSTOMER_POINT	CUSTOMER_TIKETING_NUMBER
1	heebin	오희빈	1234	01072373239	충북청주시분평동	93/02/04	300	3
2	sumin	제갈수민	abcd	01024351234	대전광역시궁동	91/12/23	0	3

7.3.2. Payment

	PAYMENT_NU...	USED_POINT	CASH	CUSTOMER_ID	RESERVATION_NUMBER	TICKETING
1	16	0	30000	heebin	28	1

7.3.3. Reservation

	RESERVATION_NUMBER	SCREE...	CUSTOMER_ID
1	28	16	heebin
2	30	13	sumin

7.3.4. Ticket

	TICKET_NUMBER	SEAT_NUMBER	RESERVATION_NUMBER	CINEMA_NAME	SCREEN_NUMBER	SCREENING_NUMBER
1	40	1	28	롯데시네마청주점	1	16
2	41	2	28	롯데시네마청주점	1	16
3	42	6	28	롯데시네마청주점	1	16
4	44	3	30	CGV유성온천	1	13
5	45	8	30	CGV유성온천	1	13

7.3.5. Screening

	SCREENING_TIME	MOVIE_ID	CINEMA_NAME	SCREEN_NUMBER	SCREENING_NUMBER	REMAIN_SEAT_NUMBER
1	16/12/09	a2	CGV유성온천	1	12	10
2	16/12/09	a2	CGV유성온천	1	13	8
3	16/12/19	a1	CGV유성온천	2	14	5
4	16/12/12	a3	롯데시네마청주점	1	16	3
5	16/12/09	a1	롯데시네마청주점	3	17	4

7.3.6. Movie

	MOVIE_ID	MOVIE_NAME	RESERVATION_RATE	RUNINGTIME	INFORMATION	GRADE	DIRECTOR	MOVIE_CAST
1	a2	형	40	112	사기꾼 형과 운동선수 동생의 코미디 영화	전체	권수경	조정석,디오,박신혜
2	a1	판도라	25	136	최대 규모의 강진에 이어 원자력 폭발 사고...	12	박정우	김남길,김영애,문정희
3	a3	라라랜드	30	127	꿈을 꾸는 사람들을 위한 별들의 도시 라라...	12	다미엔 차...	라이언 고슬링, 엠마 스톤

7.3.7. Cinema

	CINEMA_NAME	CINEMA_ADDRESS	CINEMA_PHONENUMBER	CINEMA_TOTAL_SEAT_NUMBER	TOTAL_SCREEN_NUMBER
1	롯데시네마청주점	청주시성안길	0432351234	14	3
2	CGV유성온천	대전광역시궁동	01023124563	15	2

7.3.8. Screen

	SCREEN_NUMBER	SCREEN_TOTAL_SEAT_NUMBER	CINEMA_NAME
1		1	6 롯데시네마청주점
2		2	4 롯데시네마청주점
3		3	4 롯데시네마청주점
4		1	10 CGV유성온천
5		2	5 CGV유성온천

7.3.9. Seat

	SEAT_NUMBER	CINEMA_NAME	SCREEN_NUMBER
1	1	CGV유성온천	1
2	1	CGV유성온천	2
3	1	롯데시네마청주점	1
4	1	롯데시네마청주점	2
5	1	롯데시네마청주점	3
6	10	CGV유성온천	1
7	2	CGV유성온천	1
8	2	CGV유성온천	2
9	2	롯데시네마청주점	1
10	2	롯데시네마청주점	2
11	2	롯데시네마청주점	3
12	3	CGV유성온천	1
13	3	CGV유성온천	2
14	3	롯데시네마청주점	1
15	3	롯데시네마청주점	2
16	3	롯데시네마청주점	3
17	4	CGV유성온천	1
18	4	CGV유성온천	2
19	4	롯데시네마청주점	1

8. 응용 프로그램 함수

8.1. Customer Class

8.1.1. login

```
//회원의 ID를 입력받고 테이블에 존재하면 password를 확인하고 같은 튜블에 존재하면 로그인을 한다.
	public boolean login(Connection conn) {
		String userPW = "";
		String inputPW;

		System.out.print("What is your ID? : ");
		inputtedUserID = scan.nextLine();
		System.out.print("What is your pw? : ");
		inputPW = scan.nextLine();

		try {
			ResultSet rs = this.select(conn,
					"SELECT CUSTOMER_PW FROM CUSTOMER WHERE
CUSTOMER_ID = '" + inputtedUserID + "'");
			rs.next();
			userPW = rs.getString(1);
			if (inputPW.equals(userPW)) {
				System.out.println("Log-in success!");
				return true;
			} else {
				System.out.println("Fail to Log-in");
				return false;
			}
		} catch (Exception e) {
			System.out.println("There is not your ID");
			// System.out.println("[*] 질의 결과 출력 오류 발생: \n" + e.getMessage());
			return false;
		}

	}
```

8.1.2. sign up

```
	//고객의 회원가입 ID의 중복을 확인하고 테이블에 존재하지 않는 경우에는 회원가입을 진행한다.
	public boolean signUp(Connection conn) {
		String inputID = "";
		String inputName, inputPW, inputPhoneNumber, inputAddress, inputBirth;
		boolean nobodyUseThisID = false;
		while (nobodyUseThisID == false) {
			System.out.print("ID : ");
			inputID = scan.nextLine();
			//고객 테이블에 회원 ID의 유무를 확인한다.
			try {
				ResultSet rs = this.select(conn,
						"SELECT CUSTOMER_ID FROM CUSTOMER WHERE
CUSTOMER_ID = '" + inputID + "'");
				rs.next();
				rs.getString(1);
				System.out.println("You cannot use this ID.");
				System.out.println("Please input another ID.");

			} catch (Exception e) {
				System.out.println("You can use this ID.");
				nobodyUseThisID = true;
			}
		}
		System.out.print("Name : ");
		inputName = scan.nextLine();
		System.out.print("PW : ");
		inputPW = scan.nextLine();
		System.out.print("Phone number : ");
		inputPhoneNumber = scan.nextLine();
		System.out.print("Address : ");
		inputAddress = scan.nextLine();
		System.out.print("Birth(e.g. 93/07/12) : ");
		inputBirth = scan.nextLine();

		return this.insert(conn, "INSERT INTO CUSTOMER VALUES('" + inputID + "','" + inputName
+ "','" + inputPW + "','"
				+ inputPhoneNumber + "','" + inputAddress + "','" + inputBirth +
"',0,0)");

	}
```

8.1.3. booking movie

```
//예매를 하는 함수 결제 전의 과정까지 진행한다.
	public void bookingMovie(Connection conn) {
		String movieId, movieName, screenNum = "";
		String inputCinemaName, inputMovieName, inputSeatNum;
		int reservationSeatCount = 0, remainSeat, screenTotalSeatNumber, reservationNumber = 0;
		String inputScreeningNumber;
		//영화관의 이름들을 출력시켜서 영화관을 선택한다.
		try {
			ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
			System.out.println("=============================================");
			while (rs.next()) {
				System.out.println("영화관 이름 : " + rs.getString(1));
			}

		} catch (Exception e) {
			System.out.println("영화관이 존재하지 않습니다.");
		}
		System.out.println("-----------------------------------------------------");
		System.out.print("영화관 선택 : ");
		inputCinemaName = scan.nextLine();
		// 영화관을 선택하면 그 영화관에서 상영중인 영화를 보여준다.
		try {
			ResultSet rs = this.select(conn, "SELECT MOVIE_ID, MOVIE_NAME FROM MOVIE
WHERE MOVIE_ID IN ("
					+ "SELECT MOVIE_ID FROM SCREENING WHERE
CINEMA_NAME = '" + inputCinemaName + "')");
			System.out.println("-----------------------------------------------------");
			while (rs.next()) {
				movieId = rs.getString(1);
				movieName = rs.getString(2);
				System.out.println("MOVIE_ID : " + movieId + "\t MOVIE_NAME : " +
movieName);
			}
		} catch (Exception e) {
			System.out.println("그 영화관에는 영화가 상영하지 않습니다.");
		}
		System.out.println("-----------------------------------------------------");
		// 영화관에서 상영중인 영화를 선택하면 영화 상영일정들을 보여주고 상영 번호를 선택한다.
		System.out.print("영화 선택 : ");
		inputMovieName = scan.nextLine();
		try {
			ResultSet rs = this.select(conn,
					"SELECT SCREENING_TIME, MOVIE_ID, SCREENING_NUMBER,
SCREEN_NUMBER FROM SCREENING WHERE CINEMA_NAME = '"+inputCinemaName+"' AND MOVIE_ID = ("
						+ "SELECT MOVIE_ID FROM MOVIE WHERE
MOVIE_NAME = '" + inputMovieName + "')");
			System.out.println("-----------------------------------------------------");
			System.out.println("선택한 영화 : " + inputMovieName);
			rs.next();
			do {
				screenNum = rs.getString(4);
				System.out.println("상영번호 : " + rs.getString(3) + "\t상영일정 : " +
rs.getString(1));
			} while (rs.next());
		} catch (Exception e) {
			System.out.println("선택한 영화가 잘 못 됐습니다.");
		}
		System.out.println("-----------------------------------------------------");

		System.out.print("상영번호 선택 : ");
		inputScreeningNumber = scan.nextLine();

		// 예매할 매수를 선택한다.
		try {
			//선택한 영화관의 상영관 정보 중 상영관의 전체 좌석 수를 받아온다.
			ResultSet screen = this.select(conn, "SELECT SCREEN_TOTAL_SEAT_NUMBER
FROM SCREEN WHERE SCREEN_NUMBER = "
					+ screenNum + " AND CINEMA_NAME = '" + inputCinemaName
+ "'");
			screen.next();
			screenTotalSeatNumber = Integer.parseInt(screen.getString(1));
			//선택한 상영관의 남은 좌석 정보를 받아온다.
			ResultSet rs = this.select(conn,
					"SELECT REMAIN_SEAT_NUMBER FROM SCREENING WHERE
SCREENING_NUMBER = " + inputScreeningNumber);
			rs.next();
			remainSeat = Integer.parseInt(rs.getString(1));//상영관의 남은 좌석을 저장하는 변수
			System.out.print("예매할 매수 선택(남은 좌석 수 : " + remainSeat + ") : ");
			reservationSeatCount = scan.nextInt();
			scan.nextLine();
			int temp = reservationSeatCount;
			// 예매한 매수만큼 좌석을 선택한다. 이미 예매된 좌석에 대한 정보를 출력시키고 예매한
수만큼 좌석을 선택한다.
			System.out.println("-----------------------------------------------------");
			if (reservationSeatCount <= remainSeat) {
				int[] checkRemainSeat = new int[100];//예매된 좌석에 대한 정보를
확인하기위한 변수
				for (int i = 0; i < reservationSeatCount;) {
					System.out.println("좌석선택<1~" + screenTotalSeatNumber + ">
");//전체 좌석수를 보여 준다.
					try {
```

```
                                                ResultSet rt = this.select(conn,
                                                                "SELECT SEAT_NUMBER FROM
TICKET WHERE SCREENING_NUMBER = " + inputScreeningNumber);
                                                rt.next();
                                                do {

checkRemainSeat[Integer.parseInt(rt.getString(1))] = 1;
                                                } while (rt.next());
                                                //예매된 좌석에 대한 정보를 보여준다.
                                                System.out.print("<예매된 좌석(");
                                                for (int j = 0; j < checkRemainSeat.length; j++) {
                                                        if (checkRemainSeat[j] == 1)
                                                                System.out.print(j + " ");
                                                }
                                                //좌석 선택후 남은 예매 수에 대해 보여준다.
                                                System.out.print("), 남은 예매 수(" +
(reservationSeatCount - i) + ")> : ");
                                        } catch (Exception e) {
                                                System.out.print("<남은 예매 수(" +
(reservationSeatCount - i) + ")> : ");
                                        }
                                        inputSeatNum = scan.nextLine();//좌석을 선택
                                        //예매 테이블을 insert하고 티켓 테이블에 입력한 값들을 insert한다
그리고 여러장을 예매했다면 다시 좌석 선택으로 돌아간다.
                                        //고객 테이블에 존재하는 ticketing수를 1증가 시키고 상영 테이블의
남은 좌석을 1 감소 시킨다.
                                        try {
                                                try {
                                                        if (reservationNumber == 0) {
                                                                insert(conn, "INSERT INTO
RESERVATION VALUES (RESERVATENUM.NEXTVAL,"
                                                                                +
inputScreeningNumber + ",'" + inputtedUserID + "')");
                                                                ResultSet ru = this.select(conn,
"SELECT RESERVATENUM.CURRVAL FROM DUAL");
                                                                ru.next();
                                                                reservationNumber =
Integer.parseInt(ru.getString(1));
                                                        }
                                                } catch (Exception e) {
                                                        System.out.println("예약테이블시퀀스 오류");
                                                }
                                                insert(conn,
                                                                "INSERT INTO TICKET VALUES
(TICKETNUM.NEXTVAL," + inputSeatNum + "," + reservationNumber
                                                                                + ",'" +
inputCinemaName + "'," + screenNum + "," + inputScreeningNumber + ")");
                                                update(conn,
                                                                "UPDATE CUSTOMER SET
CUSTOMER_TIKETING_NUMBER = CUSTOMER_TIKETING_NUMBER + 1 WHERE CUSTOMER_ID = '"
                                                                                +
inputtedUserID + "'");
                                                update(conn,
                                                                "UPDATE SCREENING SET
REMAIN_SEAT_NUMBER = REMAIN_SEAT_NUMBER - 1 WHERE SCREENING_NUMBER = "
                                                                                +
inputScreeningNumber);
                                                i++;
                                        } catch (Exception e) {
                                                System.out.println("티켓테이블 오류");
                                        }
                                }
                                this.doPayment(conn, reservationSeatCount, reservationNumber);//결제
function을 불러온다.
                        } else {
                                System.out.println("남은 좌석이 부족합니다.");
                        }
                } catch (Exception e) {
                        System.out.println("그 일정에는 영화 상영이 없습니다.");
                }
                System.out.println("================================================");
        }
```

8.1.4. payment

```
//결제 선택 예약이 끝나고 난후 실행
	private void doPayment(Connection conn, int numOfTicket, int reservationNum) {
		int inputMethodToPayment = 0;
		int point;
		int usedPoint = 0;//사용된 포인트
		int totalCost = 10000 * numOfTicket;//예매의 총가격 한장의 티켓당 10000원
		//현장 결제인지 인터넷 결제인지 선택하고 인터넷 결제이면 결제를 실행 현장 결제이면
manager를 통해 결제를 하게 한다.
		while (inputMethodToPayment != 1 && inputMethodToPayment != 2) {
			System.out.print("결제하는 방법은? (1: 현장 결제, 2: 인터넷 결제) : ");
			inputMethodToPayment = scan.nextInt();
			scan.nextLine();
			switch (inputMethodToPayment) {
			//현장 결제인 경우 manager가 결제를 처리한다.
			case 1:
				break;
			//인터넷 결제
			case 2:
				try {
					ResultSet rs = this.select(conn,
							"SELECT CUSTOMER_POINT FROM
CUSTOMER WHERE CUSTOMER_ID = '" + inputtedUserID + "'");
					rs.next();
					point = Integer.parseInt(rs.getString(1));
				} catch (Exception e) {
					System.out.println("고객 테이블 에러");
					break;
				}
				System.out.println("인터넷 결제를 시작합니다.");
				System.out.println("당신의 총 결제 금액은 " + totalCost + "입니다.");
				System.out.println("당신의 포인트는 " + point + "p 있습니다.");
				//포인트가 1000점인 경우에만 포인트를 사용하게 한다.
				if (point >= 1000) {
					int tempInput = 0;
					while (tempInput != 1 && tempInput != 2) {
						System.out.print("포인트를 사용하시겠습니까?
(1:Yse, 2:No) : ");
						tempInput = scan.nextInt();
						scan.nextLine();
						switch (tempInput) {
						case 1:
							boolean tempUseCheck = false;
							while (tempUseCheck == false) {
								System.out.print("얼마의
포인트를 사용하시겠습니까? : ");
								usedPoint = scan.nextInt();
								scan.nextLine();
								//고객 테이블에 존재하는
point와 사용할 포인트를 비교해서 초과하지 않게 처리한다.
								if (usedPoint <= point) {
									tempUseCheck =
true;
								} else {
System.out.println("사용할 수 있는 포인트를 초과했습니다. 다시 입력하세요");
								}
							}
							break;
						case 2:
							break;
						default:
							System.out.println("Wrong input. Please
input again.");
							break;
						}
					}
				} else {
					System.out.println("포인트 부족으로 결제에는 사용하실 수
없습니다. ");
				}
				//총 가격과 포인트사용량을 체크해서 총 지불해야할 가격을 지정한다.
				int cash = totalCost - usedPoint;
				//PAYMENT테이블에 값들을 insert한다 마지막이 0인 이유는 발권 여부를
확인해주기 위해 만든 속성이다.
				this.insert(conn, "INSERT INTO PAYMENT VALUES(
PAYMENTNUM.NEXTVAL," + usedPoint + "," + cash + ",'"
						+ inputtedUserID + "'," + reservationNum + ",0)");
				int addPoint = numOfTicket * 100;//티켓한장당 100의 point를 준다.
				//포인트 정보를 고객 테이블에 update한다.
				this.update(conn, "UPDATE CUSTOMER SET CUSTOMER_POINT =
CUSTOMER_POINT+" + addPoint
						+ "WHERE CUSTOMER_ID = '" + inputtedUserID +
"'");
				System.out.println("결제에 성공했습니다!");
				break;
			default:
				System.out.println("Wrong input!!!");
				break;
			}
		}
	}
```

8.1.5. search movie

```
//예매율에 대한 영화 차트를 보여준다.
	public void SearchMovie(Connection conn) {
		try {
			//예매율에 따라 영화 정보를 높은 순으로 보여준다.
			ResultSet rs = this.select(conn,
					"SELECT MOVIE_NAME, RESERVATION_RATE
FROM MOVIE ORDER BY RESERVATION_RATE DESC");

System.out.println("================================================");
			System.out.printf("%30s		|%10s\n", "영화 제목", "예매율");

System.out.println("------------------------------------------------");
			rs.next();
			do {
				System.out.printf("%30s		|%10s%s\n",
rs.getString(1), rs.getString(2), "%");
			} while (rs.next());

System.out.println("================================================");
			System.out.println();
		} catch (Exception e) {
			System.out.println("There is no movie.");
		}
	}
```

8.1.6. modify information

```
//회원 정보를 수정하는 function
	public void modifyMyInformation(Connection conn) {
		try {
			int inputForModify = 0;

			System.out.println(inputtedUserID);//현재 로그인된 회원의 ID
			ResultSet rs = this.select(conn, "SELECT * FROM CUSTOMER WHERE
CUSTOMER_ID = '" + inputtedUserID + "'");
			rs.next();
			//현재 로그인된 회원의 정보를 출력시켜준다.

System.out.println("=======================================================");
			System.out.println("This is your information");

System.out.println("-------------------------------------------------------");
			System.out.println("ID : " + rs.getString(1));
			System.out.println("Name : " + rs.getString(2));
			System.out.println("Password : " + rs.getString(3));
			System.out.println("Phone number : " + rs.getString(4));
			System.out.println("Adderss : " + rs.getString(5));
			System.out.println("Birth : " + rs.getString(6));
			System.out.println("Point : " + rs.getString(7));

System.out.println("=======================================================");
			//수정하고자 하는 부분을 선택해 수정을 한다.
			while (inputForModify != 6) {
				System.out.print("What do you want to change? (1: name, 2:
password, 3: phone number,"
						+ " 4: address, 5: birth, 6: exit)");
				inputForModify = scan.nextInt();
				scan.nextLine();
				String userInput = "";
				if (inputForModify != 5 && inputForModify != 6) {
					System.out.print("How do you want to change? : ");
					userInput = scan.nextLine();
				}
				switch (inputForModify) {
				case 1:
					this.update(conn, "UPDATE CUSTOMER SET
CUSTOMER_NAME = '" + userInput + "' WHERE CUSTOMER_ID = '"
							+ inputtedUserID + "'");
					break;
				case 2:
					this.update(conn, "UPDATE CUSTOMER SET
CUSTOMER_PW = '" + userInput + "' WHERE CUSTOMER_ID = '"
							+ inputtedUserID + "'");
					break;
				case 3:
					this.update(conn, "UPDATE CUSTOMER SET
CUSTOMER_PHONENUMBER = '" + userInput
							+ "' WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
					break;
				case 4:
					this.update(conn, "UPDATE CUSTOMER SET
CUSTOMER_ADDRESS = '" + userInput + "' WHERE CUSTOMER_ID = '"
```

```
                                                                + inputtedUserID + "'");
                                                        break;
                                                case 5:
                                                        System.out.print("How do you want to change?(e.g.
93/07/12) : ");
                                                        userInput = scan.nextLine();
                                                        this.update(conn, "UPDATE CUSTOMER SET
CUSTOMER_BRITH = '" + userInput + "' WHERE CUSTOMER_ID = '"
                                                                + inputtedUserID + "'");
                                                        break;
                                                case 6:
                                                        break;
                                                default:
                                                        System.out.println("Wrong input!!!");
                                                        break;
                                                }
                                        }

                                } catch (Exception e) {
                                        System.out.println("Wrong ID.");
                                }
                        }
```

8.1.7. check reservation

```
//현재 로그인된 회원에 대한 예매 정보, 예매 삭제를 하는 function
        public void checkMyReservation(Connection conn) {
                int indexForReservation = 0;
                try {
                        //로그인된 회원이 예매한 예약번호, 영화제목, 영화관, 상영시간, 상영관을 보여준다.
                        ResultSet rs = this.select(conn,
                                        "SELECT MOVIE_ID, RESERVATION_NUMBER,
CINEMA_NAME, SCREENING_TIME, SCREEN_NUMBER FROM RESERVATION, SCREENING WHERE
RESERVATION.CUSTOMER_ID = '"
                                                        + inputtedUserID + "' AND
SCREENING.SCREENING_NUMBER = RESERVATION.SCREENING_NUMBER");
                        System.out.println(

"=============================================================================================="
);
                        System.out.printf("%10s        %20s        %10s        %15s        %5s\n",
"예약번호", "영화 제목", "영화관", "상영 시간",
                                        "상영관");
                        System.out.println(

"----------------------------------------------------------------------------------------------"
);
                        rs.next();
                        do {
                                ResultSet rt = this.select(conn,
                                                "SELECT MOVIE_NAME FROM MOVIE WHERE
MOVIE_ID = '" + rs.getString(1) + "'");
                                rt.next();
                                System.out.printf("%10s        %20s        %10s        %15s
%5s\n", rs.getString(2),
                                                rt.getString(1), rs.getString(3), rs.getString(4),
rs.getString(5));
                        }while (rs.next());
                        System.out.println(

"=============================================================================================="
);
                        System.out.println();
                } catch (Exception e) {
                        System.out.println("There is no reservation.");
                }
                //예매에 대해서 몇장의 티켓을 예매했는지와 예매 취소를 선택한다.
                while (indexForReservation != 3) {
                        System.out.println(
                                        "What do you want to do? (1: show tickets of reservation,
2: delete reservation, 3: exit) : ");
                        indexForReservation = scan.nextInt();
                        scan.nextLine();
                        String inputReservationNumber;
                        switch (indexForReservation) {
                        //한번의 예매에서 몇장의 티켓을 샀는지 보여준다.
                        case 1:
                                System.out.println("What reservation do you want to see more?
(Input the reservation number) : ");
                                inputReservationNumber = scan.nextLine();
                                try {
                                        ResultSet rs = this.select(conn,
                                                        "SELECT * FROM TICKET WHERE
RESERVATION_NUMBER = " + inputReservationNumber);
                                        System.out.println(

"=============================================================================================="
);
                                        System.out.printf("%10s        %10s        %10S        %10S
        %5S\n", "티켓번호", "좌석번호", "예약번호",
```

```
                                                                "영화관", "상영관");
                                                        System.out.println(
"-------------------------------------------------------------------------------------------------------"
);
                                                        while (rs.next()) {
                                                                System.out.printf("%10s        %10s        %10S
    %10S        %5S\n", rs.getString(1),
                                                                                rs.getString(2),
rs.getString(3), rs.getString(4), rs.getString(5));
                                                        }
                                                        System.out.println(
"======================================================================================================="
);
                                                        System.out.println();
                                                } catch (Exception e) {
                                                        System.out.println("You input wrong number");
                                                }
                                                break;
                                        //예매 취소
                                        case 2:
                                                String screeningNumber = "";
                                                System.out.println("What reservation do you want to delete? (Input
the reservation number) : ");
                                                inputReservationNumber = scan.nextLine();
                                                //취소할 예매 번호를 선택하고 선택한 예매번호와 일치하는 예매 테이블과
티켓 테이블의 정보를 삭제한다.
                                                //예매 취소를 하면 point를 티켓수*100만큼 감소 시키고 고객의
티켓팅수를 감소 시킨다.
                                                try {
                                                        ResultSet rs = this.select(conn, "SELECT
SCREENING_NUMBER FROM TICKET WHERE RESERVATION_NUMBER = "
                                                                        +
Integer.parseInt(inputReservationNumber));
                                                        int ticketCount = 0;
                                                        while (rs.next()) {
                                                                ticketCount++;
                                                                screeningNumber = rs.getString(1);
                                                        }
                                                        delete(conn, "DELETE FROM RESERVATION WHERE
RESERVATION_NUMBER = "
                                                                        +
Integer.parseInt(inputReservationNumber));
                                                        update(conn, "UPDATE SCREENING SET
REMAIN_SEAT_NUMBER = REMAIN_SEAT_NUMBER +" + ticketCount
                                                                        + " WHERE SCREENING_NUMBER = '" +
screeningNumber + "'");
                                                        update(conn, "UPDATE CUSTOMER SET CUSTOMER_POINT
= CUSTOMER_POINT -" + (ticketCount * 100)
                                                                        + " WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
                                                        update(conn, "UPDATE CUSTOMER SET
CUSTOMER_TIKETING_NUMBER = CUSTOMER_TIKETING_NUMBER -" + ticketCount
                                                                        + " WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
                                                } catch (Exception e) {
                                                        System.out.println("You input wrong number");
                                                }
                                                break;
                                        case 3:
                                                break;
                                        default:
                                                System.out.println("Wrong input!!!");
                                                break;
                                        }
                                }
                        }
```

8.1.8. secession

```
//회원 탈퇴 function
        public void secession(Connection conn) {
                int inputForsecession;
                System.out.print("Are you sure to remove your account? (1: Yes, 2: No!) : ");
                inputForsecession = scan.nextInt();
                scan.nextLine();
                switch (inputForsecession) {
                //현재 로그인된 고객에 대한 튜플을 고객 테이블에서 삭제 시킨다.
                case 1:
                        this.delete(conn, "DELETE FROM Customer WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
                        break;
                case 2:
                        break;
                default:
                        System.out.println("Wrong input!!!");
                        break;
                }
        }
```

8.2. Manager Class

8.2.1. manage movie

```
// 영화 등록, 수정, 삭제를 하는 함수
    public void manageMovie(Connection conn) {
        int inputForManageMovie = 0;
        String movieId;
        String name, id;
        String MovieId, MovieName, information, director, grade, MovieCast;
        int reservationRate, runingtime;

        while (inputForManageMovie != 4) {
            System.out.print("What do you want to do? (1: 영화 등록, 2: 영화 삭제, 3:
영화 정보 수정, 4: exit) : ");
            inputForManageMovie = scan.nextInt();
            scan.nextLine();

            switch (inputForManageMovie) {
            // 영화등록 영화ID, 영화제목, 러닝타임, 영화정보, 등급, 감독, 배우를 입력해서
테이블에 저장시킨다.
            case 1:

System.out.println("=================================================");
                System.out.print("영화ID? : ");
                MovieId = scan.nextLine();
                System.out.print("영화제목? : ");
                MovieName = scan.nextLine();
                System.out.print("RuningTime? : ");
                runingtime = scan.nextInt();
                scan.nextLine();
                System.out.print("영화정보? : ");
                information = scan.nextLine();
                System.out.print("GRADE?(전체, 12, 15, 19) : ");
                grade = scan.nextLine();
                System.out.print("감독? : ");
                director = scan.nextLine();
                System.out.print("배우? : ");
                MovieCast = scan.nextLine();

System.out.println("=================================================");
                insert(conn, "INSERT INTO MOVIE VALUES('" + MovieId + "','" +
MovieName + "'," + 0 + "," + runingtime
                        + ",'" + information + "','" + grade + "','" + director + "','" +
MovieCast + "')");
                break;
            // 등록된 영화 삭제 영화ID를 입력해서 등록된 영화정보를 삭제한다.
            case 2:

System.out.println("=================================================");
                try {
                    ResultSet rs = this.select(conn, "SELECT MOVIE_NAME, MOVIE_ID
FROM MOVIE");
                    rs.next();
                    do {
                        name = rs.getString(1);
                        id = rs.getString(2);
                        System.out.println("ID : " + id + " MOVIE_NAME : " + name);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("등록된 영화가 없습니다. ");
                }

System.out.println("=================================================");
                System.out.print("삭제할 영화ID? : ");
                movieId = scan.nextLine();
                // 삭제할 영화ID를 입력해서 테이블에 존재하면 DELETE 존재하지 않은
경우에는 오류를 출력시킨다.
                try {
                    ResultSet rs = this.select(conn, "SELECT MOVIE_ID FROM MOVIE
WHERE MOVIE_ID = '" + movieId + "'");
                    rs.next();
```

```
                        id = rs.getString(1);
                        if (movieId.equals(id)) {
                            delete(conn, "DELETE FROM MOVIE WHERE MOVIE_ID = '" +
movieId + "'");
                            System.out.println("delete success!");
                        } else {
                            System.out.println("Fail to delete");
                        }
                    } catch (Exception e) {
                        System.out.println("삭제할 영화가 없습니다.");
                    }
                    break;
                // 영화정보에 대해서 수정한다. 수정할 영화 ID를 선택해서 영화 테이블에
존재하는 영화 ID를 제외하고 나머지 정보들을
                // 수정할 수 있다.
                case 3:
                    int inputForModifyMovieInformation = 0;

System.out.println("================================================");
                    try {
                        ResultSet rs = this.select(conn, "SELECT MOVIE_NAME, MOVIE_ID
FROM MOVIE");
                        rs.next();
                        do {
                            name = rs.getString(1);
                            id = rs.getString(2);
                            System.out.println("ID : " + id + " MOVIE_NAME : " + name);
                        } while (rs.next());
                    } catch (Exception e) {
                        System.out.println("[*] 질의 결과 출력 오류 발생: \n" +
e.getMessage());
                    }

System.out.println("================================================");
                    System.out.print("수정할 영화ID? : ");
                    movieId = scan.nextLine();
                    // 입력한 영화ID와 같은 영화ID가 테이블에 존재할 경우 영화 정보를
수정하고 없다면 오류를 출력시킨다.
                    try {
                        ResultSet rs = this.select(conn, "SELECT MOVIE_ID FROM MOVIE
WHERE MOVIE_ID = '" + movieId + "'");
                        rs.next();
                        id = rs.getString(1);
                        // 영화 정보 수정 선택
                        if (movieId.equals(id)) {
                            while (inputForModifyMovieInformation != 8) {
                                System.out
                                        .print("무엇을 수정하시겠습니까? (1: MOVIE_NAME, 2:
RESERVATION_RATE, 3: RUNINGTIME, 4: INFORMATION"
                                                + ", 5: GRADE, 6: DIRECTOR, 7:
MOVIE_CAST, 8: exit) : ");
                                inputForModifyMovieInformation = scan.nextInt();
                                scan.nextLine();
                                switch (inputForModifyMovieInformation) {
                                case 1:
                                    System.out.print("수정할 MOVIE_NAME? : ");
                                    MovieName = scan.nextLine();
                                    update(conn, "UPDATE MOVIE SET MOVIE_NAME = '" +
MovieName + "' WHERE MOVIE_ID = '"
                                            + movieId + "'");
                                    break;
                                case 2:
                                    System.out.print("수정할 RESERVATION_RATE?(100이하)
: ");
                                    reservationRate = scan.nextInt();
                                    scan.nextLine();
                                    update(conn, "UPDATE MOVIE SET
RESERVATION_RATE = " + reservationRate
                                            + " WHERE MOVIE_ID = '" + movieId + "'");
                                    break;
                                case 3:
                                    System.out.print("수정할 RUNINGTIME? : ");
                                    runingtime = scan.nextInt();
                                    scan.nextLine();
```

```
                                    update(conn, "UPDATE MOVIE SET RUNINGTIME = " +
runingtime + " WHERE MOVIE_ID = '"
                                            + movieId + "'");
                                    break;
                                case 4:
                                    System.out.print("수정할 INFORMATION? : ");
                                    information = scan.nextLine();
                                    update(conn, "UPDATE MOVIE SET INFORMATION = '"
+ information + "' WHERE MOVIE_ID = '"
                                            + movieId + "'");
                                    break;
                                case 5:
                                    System.out.print("수정할 GRADE?(전체, 12, 15, 19) : ");
                                    grade = scan.nextLine();
                                    update(conn,
                                            "UPDATE MOVIE SET GRADE = '" + grade + "'
WHERE MOVIE_ID = '" + movieId + "'");
                                    break;
                                case 6:
                                    System.out.print("수정할 DIRECTOR? : ");
                                    director = scan.nextLine();
                                    update(conn, "UPDATE MOVIE SET DIRECTOR = '" +
director + "' WHERE MOVIE_ID = '"
                                            + movieId + "'");
                                    break;
                                case 7:
                                    System.out.print("수정할 MOVIE_CAST? : ");
                                    MovieCast = scan.nextLine();
                                    update(conn, "UPDATE MOVIE SET MOVIE_CAST = '" +
MovieCast + "' WHERE MOVIE_ID = '"
                                            + movieId + "'");
                                    break;
                                case 8:
                                    break;
                                default:
                                    System.out.println("Wrong input!!!");
                                    break;
                                }
                            }
                        }
                    } catch (Exception e) {// 입력한 영화ID가 DB에 존재하지 않는경우
                        System.out.println("입력한 영화ID가 존재하지 않습니다.");
                    }
                    break;
                case 4:
                    break;
                default:
                    System.out.println("Wrong input!!!");
                    break;
                }
            }

        }
```

8.2.2. manage cinema

```
// 영화관 정보 등록, 삭제, 수정
    public void manageCinema(Connection conn) {
        int inputForManageCinema = 0;
        String name, cinemaName, cinemaAddress, cinemaPhonenumber;
        int cinemaTotalSeatNumber, cinemaScreenNumber;
        while (inputForManageCinema != 4) {
            System.out.print(
                    "What do you want to do? (1: Upload cinema, 2: delete cinema, 3: modify cinema
information, 4: exit) : ");
            inputForManageCinema = scan.nextInt();
            scan.nextLine();
            System.out.println("==================================================");
            switch (inputForManageCinema) {
            case 1:// 영화관등록
                System.out.print("영화관이름? : ");
                cinemaName = scan.nextLine();
                System.out.print("영화관주소? : ");
                cinemaAddress = scan.nextLine();
                System.out.print("영화관전화번호? : ");
                cinemaPhonenumber = scan.nextLine();
                System.out.print("영화관총좌석수? : ");
                cinemaTotalSeatNumber = scan.nextInt();
                scan.nextLine();
                System.out.print("영화관 전체 상영관수? : ");
                cinemaScreenNumber = scan.nextInt();
                scan.nextLine();

                this.insert(conn, "INSERT INTO CINEMA VALUES('" + cinemaName + "','" + cinemaAddress +
"','"
                        + cinemaPhonenumber + "'," + cinemaTotalSeatNumber + "," + cinemaScreenNumber
+ ")");
                System.out.println("==================================================");
                break;

            case 2:// 영화관삭제

                try {
                    ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
                    rs.next();
                    do {
                        name = rs.getString(1);
                        System.out.println("CinemaName : " + name);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("[*] 질의 결과 출력 오류 발생: \n" + e.getMessage());
                }

                System.out.print("삭제할 영화관이름? : ");
                cinemaName = scan.nextLine();

                try {
                    ResultSet rs = this.select(conn,
                            "SELECT CINEMA_NAME FROM CINEMA WHERE CINEMA_NAME = '" +
cinemaName + "'");
                    rs.next();
                    name = rs.getString(1);
                    if (cinemaName.equals(name)) {
                        delete(conn, "DELETE FROM CINEMA WHERE CINEMA_NAME = '" + cinemaName +
"'");

                        System.out.println("delete success!");
                    } else {
                        System.out.println("Fail to delete");
                    }
                } catch (Exception e) {
                    System.out.println("삭제할 영화관이 존재하지 않습니다.");
                }
                break;

            case 3:// 영화관수정
                int inputForModifyCinemaInformation = 0;
                try {
                    ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
                    rs.next();
                    do {
                        name = rs.getString(1);
                        System.out.println("CINEMA_NAME : " + name);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("[*] 질의 결과 출력 오류 발생: \n" + e.getMessage());
                }

                System.out.print("수정할 영화관이름? : ");
                cinemaName = scan.nextLine();
                // 수정할 영화관을 선택하고 없으면 오류를 출력시킨다.
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT CINEMA_NAME FROM CINEMA WHERE CINEMA_NAME = '" +
cinemaName + "'");
                    rs.next();
                    name = rs.getString(1);
                    if (cinemaName.equals(name)) {
```

```
                        // 수정할 것을 선택한다.
                        while (inputForModifyCinemaInformation != 5) {
                            System.out
                                    .print("무엇을 수정하시겠습니까? (1: CINEMA_ADDRESS, 2:
CINEMA_PHONENUMBER , 3: CINEMA_TOTAL_SEAT_NUMBER "
                                            + "4: TOTAL_SCREEN_NUMBER, 5: exit) : ");
                            inputForModifyCinemaInformation = scan.nextInt();
                            scan.nextLine();
                            switch (inputForModifyCinemaInformation) {
                            case 1:
                                System.out.print("수정할 CINEMA_ADDRESS? : ");
                                cinemaAddress = scan.nextLine();
                                update(conn, "UPDATE CINEMA SET CINEMA_ADDRESS = '" + cinemaAddress
                                        + "' WHERE CINEMA_NAME = '" + cinemaName + "'");
                                break;
                            case 2:
                                System.out.print("수정할 CINEMA_PHONENUMBER? : ");
                                cinemaPhonenumber = scan.nextLine();
                                update(conn, "UPDATE CINEMA SET CINEMA_PHONENUMBER = '" +
cinemaPhonenumber
                                        + "' WHERE CINEMA_NAME = '" + cinemaName + "'");
                                break;
                            case 3:
                                System.out.print("수정할 CINEMA_TOTAL_SEAT_NUMBER? : ");
                                cinemaTotalSeatNumber = scan.nextInt();
                                scan.nextLine();
                                update(conn, "UPDATE CINEMA SET CINEMA_TOTAL_SEAT_NUMBER = '" +
cinemaTotalSeatNumber
                                        + "' WHERE CINEMA_NAME = '" + cinemaName + "'");
                                break;
                            case 4:
                                System.out.print("수정할 TOTAL_SCREEN_NUMBER? : ");
                                cinemaScreenNumber = scan.nextInt();
                                scan.nextLine();
                                update(conn, "UPDATE CINEMA SET TOTAL_SCREEN_NUMBER = '" +
cinemaScreenNumber
                                        + "' WHERE CINEMA_NAME = '" + cinemaName + "'");
                                break;
                            case 5:
                                break;
                            default:
                                System.out.println("Wrong input!!!");
                                break;
                            }
                        }
                    }
                } catch (Exception e) {
                    System.out.println("수정할 영화관이 존재하지 않습니다.");
                }
                break;
            case 4:
                break;
            default:
                System.out.println("Wrong input!!!");
                break;
            }
        }
    }
```

8.2.3. manage screen

```
// 상영관은 지정하면 수정을 하지않고 삭제를 하지 못한다.
    public void manageScreen(Connection conn) {
        int inputForManageScreen = 0;
        String cinemaName;
        int screenNumber, screenTotalSeatNumber;
        int totalScreenNumberOfEachCinema = 0, totalSeatNumberOfEachCinema = 0, seatNum = 0,
sumSeatNum = 0;
        int num = 0, screenNum, tempScreen;
        while (inputForManageScreen != 2) {
            System.out.print("What do you want to do? (1: Upload screen, 2: exit) : ");
            inputForManageScreen = scan.nextInt();
            scan.nextLine();
            switch (inputForManageScreen) {
            case 1:// 상영관등록
                try {
                    ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        cinemaName = rs.getString(1);
                        System.out.println("CinemaName : " + cinemaName);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("[*] 질의 결과 출력 오류 발생: \n" + e.getMessage());
                }
                System.out.println("================================================");
                System.out.print("상영관을 추가할 영화관 선택 : ");
                cinemaName = scan.nextLine();
                // 상영관을 추가할 영화관이 없다면 오류를 출력한다.
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREEN_TOTAL_SEAT_NUMBER FROM SCREEN WHERE CINEMA_NAME
= '" + cinemaName + "'");
                    rs.next();
                    do {
                        seatNum = Integer.parseInt(rs.getString(1));
                        sumSeatNum = sumSeatNum + seatNum;
                        num++;
                    } while (rs.next());
                } catch (Exception e) {
                }
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT CINEMA_TOTAL_SEAT_NUMBER, TOTAL_SCREEN_NUMBER FROM
CINEMA WHERE CINEMA_NAME = '"
                                    + cinemaName + "'");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        totalSeatNumberOfEachCinema = Integer.parseInt(rs.getString(1));
                        totalScreenNumberOfEachCinema = Integer.parseInt(rs.getString(2));
                        // 영화관의 전체 좌석수를 상영관을 추가할때마다 감소 시킨다.
                        totalSeatNumberOfEachCinema = totalSeatNumberOfEachCinema - sumSeatNum;
                        System.out.println("Total Screen Number : " + totalScreenNumberOfEachCinema
                                + "\nLeft Total Seat Number : " + totalSeatNumberOfEachCinema);
                    } while (rs.next());
                    System.out.println("================================================");
                } catch (Exception e) {
                    System.out.println("[*] 질의 결과 출력 오류 발생: \n" + e.getMessage());
                }
                tempScreen = totalScreenNumberOfEachCinema - num;// 영화관의 전체
                                                                 // 상영관에서 남은
                                                                 // 상영관이 몇개인지
                                                                 // 보여주기 위해
                                                                 // 만든 변수
                int i = 0;
                int[] madeScreen = new int[totalScreenNumberOfEachCinema + 1];// 만들어진
                                                                              // 상영관을
                                                                              // 구별하기
                                                                              // 위해
                                                                              // 만든
                                                                              // 변수.
                                                                              // 만들어져있다면
                                                                              // 1로
                                                                              // 저장한다.
                while (i < tempScreen) {
                    do {
                        System.out.print("<만들어진 상영관 : ");
                        try {
                            ResultSet rs = this.select(conn,
                                    "SELECT SCREEN_NUMBER FROM SCREEN WHERE CINEMA_NAME = '" +
cinemaName + "'");
                            rs.next();
                            do {
                                screenNum = Integer.parseInt(rs.getString(1));
                                System.out.print(screenNum + " ");
                            } while (rs.next());
                        } catch (Exception e) {
                        }
                        System.out.println(">");
                        System.out.print("상영관 번호(Number) : ");
```

```
                        screenNumber = scan.nextInt();
                        scan.nextLine();
                        if (screenNumber > totalScreenNumberOfEachCinema) {// 영화관의
                                                                           // 전체
                                                                           // 좌석수를
                                                                           // 초과하지
                                                                           // 않게끔
                                                                           // 하기
                                                                           // 위해
                                                                           // 만든
                                                                           // if문
                            System.out.println("없는 상영관입니다. 다시입력하세요.");
                        } else if (madeScreen[screenNumber] != 0) {
                            System.out.println("이미 만들어져 있는 상영관입니다. 다시 입력 하세요.");
                        } else {
                            do {
                                System.out.print("원하는 좌석 수 <남은 전체 좌석수(" +
totalSeatNumberOfEachCinema + ")> : ");
                                screenTotalSeatNumber = scan.nextInt();
                                scan.nextLine();
                                if (screenTotalSeatNumber > totalSeatNumberOfEachCinema) {
                                    System.out.println("좌석이 초과 되었습니다. 다시 입력하세요");
                                }
                            } while (screenTotalSeatNumber > totalSeatNumberOfEachCinema);
                            totalSeatNumberOfEachCinema = totalSeatNumberOfEachCinema -
screenTotalSeatNumber;
                            i++;
                            madeScreen[screenNumber] = 1;
                            insert(conn, "INSERT INTO SCREEN VALUES(" + screenNumber + "," +
screenTotalSeatNumber
                                    + ",'" + cinemaName + "')");
                            for (int j = 1; j <= screenTotalSeatNumber; j++) {
                                insert(conn, "INSERT INTO SEAT VALUES('" + j + "','" + cinemaName + "'," +
screenNumber
                                        + ")");
                            }
                        }
                    } while (screenNumber > totalScreenNumberOfEachCinema);

                }
                break;

            case 2:
                break;
            default:
                System.out.println("Wrong input!!!");
                break;
            }
        }
    }
```

8.2.4. manage screening

```
// 상영은 등록과 삭제가 있다.
    public void manageScreening(Connection conn) {
        int inputForManageScreening = 0, screenTotalSeat = 0;
        String cinemaName, movieId, screeningTime, movieName;
        String screenNumber, screeningNumber;

        while (inputForManageScreening != 3) {
            System.out.print("What do you want to do? (1: 상영 정보 등록, 2: 상영 정보 삭제, 3: exit) : ");
            inputForManageScreening = scan.nextInt();
            scan.nextLine();
            switch (inputForManageScreening) {
            case 1:// 상영등록
                try {
                    ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        cinemaName = rs.getString(1);
                        System.out.println("cinemaName : " + cinemaName);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("상영할 영화관이 없습니다.");
                    break;
                }
                System.out.println("================================================");
                System.out.print("상영을 추가할 영화관 선택 : ");
                cinemaName = scan.nextLine();
                System.out.println("================================================");
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREEN_NUMBER FROM SCREEN WHERE CINEMA_NAME = '" +
cinemaName + "'");
                    rs.next();
                    do {
                        screenNumber = rs.getString(1);
                        System.out.println("screenNumber : " + screenNumber);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("이 영화관에는 상영할 상영관이 없습니다.");
                    break;
                }
                System.out.println("================================================");
                System.out.print("상영할 상영관 선택 : ");
                screenNumber = scan.nextLine();
                // 선택한 상영관이 테이블에 존재하지 않는 경우 오류 발생
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREEN_NUMBER FROM SCREEN WHERE SCREEN_NUMBER = " +
screenNumber);
                    rs.next();
                    rs.getString(1);
                } catch (Exception e) {
                    System.out.println("상영할 상영관을 잘못 선택했습니다.");
                    break;
                }
                // 상영테이블에 존재하는 그 영화관과 선택한 상영관의 상영일정을 보여준다.
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREENING_TIME, MOVIE_ID FROM SCREENING WHERE
SCREEN_NUMBER = " + screenNumber
                                    + " AND CINEMA_NAME = '" + cinemaName + "'");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        screeningTime = rs.getString(1);
                        movieId = rs.getString(2);
                        ResultSet rt = this.select(conn,
                                "SELECT MOVIE_NAME FROM MOVIE WHERE MOVIE_ID = '" + movieId + "'");
                        rt.next();
                        movieName = rt.getString(1);
                        System.out.println("movieName : " + movieName + "\t\tscreeningTime : " +
screeningTime);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("해당 상영관에는 상영일정이 없습니다.");
                }
                System.out.println("================================================");
                System.out.print("상영 일정 입력 (e.g 2013/12/12 15:20:00) : ");
                screeningTime = scan.nextLine();
                // 상영을 할 영화 선택
                try {
                    ResultSet rs = this.select(conn, "SELECT MOVIE_NAME, MOVIE_ID FROM MOVIE");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        movieName = rs.getString(1);
                        movieId = rs.getString(2);
                        System.out.println("movieName : " + movieName + "\tmovieID : " + movieId);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("상영할 영화가 없습니다.");
```

```
                    break;
                }
                System.out.println("================================================");
                System.out.print("상영할 영화ID 선택 : ");
                movieId = scan.nextLine();
                // 상영관의 정보를 받아와서 상영 테이블에 정보를 insert한다.
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREEN_TOTAL_SEAT_NUMBER FROM SCREEN WHERE
SCREEN_NUMBER = " + screenNumber
                                    + " AND CINEMA_NAME = '" + cinemaName + "'");
                    rs.next();
                    screenTotalSeat = Integer.parseInt(rs.getString(1));
                } catch (Exception e) {

                }
                insert(conn,
                        "INSERT INTO SCREENING VALUES(SNUM.NEXTVAL,TO_DATE('" + screeningTime
                                + "','YYYY/MM/DD HH24:MI:SS'),'" + movieId + "','" + cinemaName + "'," +
screenNumber
                                + "," + screenTotalSeat + ")");
                break;

            case 2:// 상영일정삭제
                try {
                    ResultSet rs = this.select(conn, "SELECT CINEMA_NAME FROM CINEMA");
                    System.out.println("================================================");
                    rs.next();
                    do {
                        cinemaName = rs.getString(1);
                        System.out.println("cinemaName : " + cinemaName);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("상영일정을 삭제할 영화관이 없습니다.");
                    break;
                }
                System.out.println("================================================");
                System.out.print("상영 일정을 삭제할 영화관 선택 : ");
                cinemaName = scan.nextLine();
                System.out.println("================================================");
                // 선택한 영화관에 상영일정이 없다면 오류를 출력시킨다.
                try {
                    ResultSet rs = this.select(conn,
                            "SELECT SCREENING_NUMBER, SCREENING_TIME, SCREEN_NUMBER FROM
SCREENING WHERE CINEMA_NAME = '"
                                    + cinemaName + "'");
                    rs.next();
                    do {
                        screeningNumber = rs.getString(1);
                        screeningTime = rs.getString(2);
                        screenNumber = rs.getString(3);
                        System.out.println("screeningNumber : " + screeningNumber + "\tscreenNumber : " +
screenNumber
                                + "관" + "\tscreeningTime : " + screeningTime);
                    } while (rs.next());
                } catch (Exception e) {
                    System.out.println("이 영화관에는 상영하는 상영관이 없습니다.");
                    System.out.println("================================================");
                    break;
                }
                System.out.println("================================================");
                System.out.print("상영 일정을 삭제할 상영 번호 선택  : ");
                screeningNumber = scan.nextLine();
                // 삭제할 상영번호를 입력 받아서 상영 일정을 삭제 시킨다.
                try {
                    delete(conn, "DELETE FROM SCREENING WHERE SCREENING_NUMBER = '" +
screeningNumber + "'");
                    System.out.println("delete success!");
                } catch (Exception e) {
                    System.out.println("삭제할 상영 번호를 잘못 입력했습니다.");
                }
                break;
            case 3:
                break;
            default:
                System.out.println("Wrong input!!!");
                break;
            }
        }
    }
```

8.2.5. manage VIP

```
// vip관리 고객이 예매한 티켓수를 많은 순서부터 10명까지 출력시킨다.
    public void manageVIP(Connection conn) {
        try {
            ResultSet rs = this.select(conn,
                    "SELECT CUSTOMER_ID, CUSTOMER_TIKETING_NUMBER FROM CUSTOMER ORDER BY
CUSTOMER_TIKETING_NUMBER DESC");
            System.out.println("================================================");
            System.out.printf("%30s       |%10s\n", "고객 ID", "티켓예매횟수");
            System.out.println("------------------------------------------------");
            rs.next();
            int i = 1;//10명까지 출력하기위해 만든 변수
            do {
                System.out.printf("%30s       |%10s\n", rs.getString(1), rs.getString(2));
                i++;
            } while (rs.next() && i < 11);
            System.out.println("================================================");
            System.out.println();
        } catch (Exception e) {
            System.out.println("There is no customer.");
        }
    }
```

8.2.6. ticketing

```
// 발권 결제를 완료한 회원의 정보가 결제 테이블에 저장되기에 결제 테이블에 저장된 것들만 발권을 할 수 있게 한다.
    public void ticketing(Connection conn) {
        String inputtedUserID = "";
        int paymentNum = 0;
        System.out.println("================================================");
        boolean checkUserID = false;
        // 고객 ID 출력 후 발권을 할 고객 ID 선택
        try {
            ResultSet rs = this.select(conn, "SELECT CUSTOMER_ID FROM CUSTOMER");
            rs.next();
            do {
                System.out.println("고객 ID : " + rs.getString(1));
            } while (rs.next());
        } catch (Exception e) {
            System.out.println("회원이 존재하지 않습니다.");
        }
        while (checkUserID != true) {
            System.out.println("================================================");
            System.out.print("발권을 원하는 고객의 ID 입력하세요. : ");
            inputtedUserID = scan.nextLine();
            try {
                ResultSet rs = this.select(conn,
                        "SELECT CUSTOMER_PW FROM CUSTOMER WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
                rs.next();
                rs.getString(1);
                checkUserID = true;
            } catch (Exception e) {
                System.out.println("없는 ID입니다. 다시 입력하세요. : ");
            }
        }
        // 결제(PAYMENT)테이블에 존재하는 결제 번호 중 발권 유무를 확인하는 TICKETING이 0인 경우 들만
출력시켜서
        // 발권을 하게 한다.
        try {
            ResultSet rs = this.select(conn,
                    "SELECT * FROM PAYMENT WHERE CUSTOMER_ID = '" + inputtedUserID + "' AND
TICKETING = 0");
            rs.next();
            do {
                System.out.println("결제 번호 : " + rs.getString(1) + ", 예약 번호 : " + rs.getString(5));
            } while (rs.next());
            System.out.print("발권을 원하는 결제 번호를 입력하세요 : ");
            paymentNum = scan.nextInt();
            scan.nextLine();
        } catch (Exception e) {
            System.out.println("결제내역이 없습니다. ");
            return;
        }
        // 발권이 완료되면 TICKETING을 1로 update하고 발권이 완료된다.
        update(conn, "UPDATE PAYMENT SET TICKETING = 1 WHERE PAYMENT_NUMBER = " +
paymentNum);
        System.out.println("결제번호 " + paymentNum + "의 발권이 완료 되었습니다.");
    }
```

8.2.7. help payment

```
// 결제 현장 결제를 선택한 고객에 대해서 결제를 해준다.
    public void helpPayment(Connection conn) {
        String inputtedUserID = "";
        int reservationNum = 0, point = 0, totalCost = 0, usedPoint = 0, ticketCount = 0;
        boolean checkUserID = false;
        // 고객들의 ID를 출력시키고 현장 결제를 선택한 고객을 선택
        try {
            System.out.println("================================================");
            ResultSet rs = this.select(conn, "SELECT CUSTOMER_ID FROM CUSTOMER");
            rs.next();
            do {
                System.out.println("고객 ID : " + rs.getString(1));
            } while (rs.next());
        } catch (Exception e) {
            System.out.println("회원이 존재하지 않습니다.");
        }
        while (checkUserID != true) {
            System.out.println("================================================");
            System.out.print("결제를 원하는 고객의 ID 입력하세요. : ");
            inputtedUserID = scan.nextLine();
            try {
                ResultSet rs = this.select(conn,
                        "SELECT CUSTOMER_PW FROM CUSTOMER WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
                rs.next();
                rs.getString(1);
                checkUserID = true;
            } catch (Exception e) {
                System.out.println("없는 ID입니다. 다시 입력하세요. : ");
            }
        }
        // 결제 테이블에 존재하지 않는 예매는 현장 결제를 선택한 고객이기에 결제 테이블에 존재하지 않고 예매
테이블에 존재하는 예매
        // 번호를 출력시킨다.
        try {
            ResultSet rs = this.select(conn, "SELECT RESERVATION_NUMBER FROM RESERVATION WHERE
CUSTOMER_ID = '"
                    + inputtedUserID + "' AND RESERVATION_NUMBER NOT IN (SELECT
RESERVATION_NUMBER FROM PAYMENT)");
            rs.next();
            do {
                System.out.println("예약 번호 : " + rs.getString(1));
            } while (rs.next());
            System.out.print("결제를 원하는 예약 번호를 입력하세요 : ");
            reservationNum = scan.nextInt();
            scan.nextLine();
        } catch (Exception e) {
            System.out.println("예약내역이 없습니다. ");
            return;
        }
        // 한 예매에 존재하는 티켓의 수를 확인하고 총가격을 측정한다.
        try {
            ResultSet rs = this.select(conn,
                    "SELECT CUSTOMER_POINT FROM CUSTOMER WHERE CUSTOMER_ID = '" +
inputtedUserID + "'");
            rs.next();
            point = Integer.parseInt(rs.getString(1));
            ResultSet rt = this.select(conn, "SELECT * FROM TICKET WHERE RESERVATION_NUMBER = " +
reservationNum);
            while (rt.next()) {
                ticketCount++;
            }
            totalCost = ticketCount * 10000;// 총 예매 가격을 나타내기 위한 변수
        } catch (Exception e) {
            System.out.println("고객 테이블 에러");
            return;
        }
        System.out.println("현장 결제를 시작합니다.");
        System.out.println("당신의 총 결제 금액은 " + totalCost + "입니다.");
        System.out.println("당신의 포인트는 " + point + "p 있습니다.");
        // 현재 고객 테이블에 존재하는 point가 1000점 이상인 경우만 point를 사용하게 한다.
        if (point >= 1000) {
            int tempInput = 0;
            while (tempInput != 1 && tempInput != 2) {
                System.out.print("포인트를 사용하시겠습니까? (1:Yse, 2:No) : ");
                tempInput = scan.nextInt();
                scan.nextLine();
                switch (tempInput) {
                case 1:
                    boolean tempUseCheck = false;
                    while (tempUseCheck == false) {
                        System.out.print("얼마의 포인트를 사용하시겠습니까? : ");
                        usedPoint = scan.nextInt();
                        scan.nextLine();
                        if (usedPoint <= point) {
                            tempUseCheck = true;
                        } else {
                            System.out.println("사용할 수 있는 포인트를 초과했습니다. 다시 입력하세요");
                        }
                    }
                    break;
```

```
                    case 2:
                        break;
                    default:
                        System.out.println("Wrong input. Please input again.");
                        break;
                    }
                }
            } else {
                System.out.println("포인트 부족으로 결제에는 사용하실 수 없습니다. ");
            }
            // 사용한 point만큼의 point를 줄이고 결제한 만큼의 point를 증가시키고 ticketing(발권유무)을 0으로
            // PAYMENT테이블에 insert한다.
            // 그리고 현재 남은 point를 고객 테이블에 update해준다.
            int cash = totalCost - usedPoint;
            this.insert(conn, "INSERT INTO PAYMENT VALUES( PAYMENTNUM.NEXTVAL," + usedPoint + "," +
cash + ",'"
                    + inputtedUserID + "'," + reservationNum + ",0)");
            int addPoint = ticketCount * 100;
            this.update(conn, "UPDATE CUSTOMER SET CUSTOMER_POINT = CUSTOMER_POINT+" + addPoint + "
WHERE CUSTOMER_ID = '"
                    + inputtedUserID + "'");
            System.out.println("결제에 성공했습니다!");
        }
```