Big Data Application - Final Project Report

Team 10 (Just do IT)

1. Team formation

- 1942051 Yudam Jo 조유담
- 1971058 Jinseo Hong 홍진서
- 1976042 Dahee Kim 김다희
- 1976297 Yulim Lee 이유림

2. project description

우리 조의 웹사이트는 상단 네비게이션 바에서 볼 수 있듯 크게 Home, Filter, Genre, DashBoard, Director / Sales / My page / Login 페이지로 나누어진다. Home, Filter의 세부 페이지로는 detail 페이지가, Login의 세부 페이지에는 Sign up 페이지가 존재한다. 또한 페이지 내용을 기준으로 '역대 한국 박스오피스 상위 100개 영화에 대한 정보 제공'과 '한국 영화 산업 통계'라는 두 개의 카테고리로 나눌 수 있다. Home, Filter, Genre, DashBoard, Director 페이지는 역대 한국 박스오피스 상위 100개 영화에 대한 정보를 제공한다. Sales 페이지는 한국 영화 산업 통계를 제공한다.

Main 페이지는 영화 제목을 입력하면 검색한 결과에 해당하는 영화의 대략적인 정보를 보여준다. Filter 페이지는 영화의 국적, 평점, 관객 수, 개봉년도를 필터링 기준으로 검색을 하면 조건에 맞는 영화 목록을 제공한다. Home 페이지와 Filter 페이지 모두 영화 제목의 링크를 클릭하면 영화의 ' detail 페이지'로 넘어간다.

Detail 페이지는 클릭한 영화의 장르, 국적, 개봉일, 관객수, 스크린 수, 감독, 배급사, 평점 등 해당 영화의 모든 정보를 제공한다. 또한 detail 페이지에서 사용자는 자신의 평점을 등록할 수 있으며, 등록한 평점의 수정, 삭제 또한 가능하다.

Genre 페이지에서 액션, SF, 드라마, 스릴러, 코미디, 애니메이션 중 사용자가 원하는 장르를 선택해 검색하면, 선택한 장르에 해당되는 영화 제목과 포스터를 보여준다.

DashBoard 페이지는 국내 역대 박스오피스의 '평점 탑 10위 영화'와 '매출 탑 10위 영화(한국, 외국영화 모두 포함)'를 '테이블'로 '관람객 수 탑 10위 영화' 와 ' 매출 탑 10위 한국 영화'를 '그래프'로 보여준다.

Director 페이지는 감독 이름을 검색하면, 해당 감독의 필모그래피를 보여준다. Director 페이지에서 헤더의 '감독정보 수정'을 클릭하면, 해당 페이지에서는 사용자가 감독의 대표작 3개를 사용자가 등록할 수 있다. 또한 '감독정보 등록'을 클릭하면 감독에 대한 정보를 추가할 수 있다. 마지막으로 '감독정보 삭제'를 클릭하면 감독의 이름을 입력받아서 감독정보를 삭제할 수 있다.

Sales 페이지는 'Sum of Yearly Sales', 'Quarterly Sales Average' , 'Film industry of all time' 이렇게 3가지 상세 페이지로 구성되어있다. 'Sum of Yearly Sales', 'Quarterly Sales Average' 페이지는 최근 5년간 한국에서 개봉된 영화의 '매출' 통계를 제공한다. 'Sum of Yearly Sales'에서는 한국 영화와 외국 영화로 구분하여 연도 별 총 매출을 보여준다. 'Quarterly Sales Average' 에서는 4분기별 별 총 매출을 보여준다. 'Film industry of all time' 페이지는 최근 5년간 한국에서 개봉된 영화를 관람한 '관람객 수' 통계를 제공한다. 이 페이지에서는 연도별, 월별, 일별 총 관람객 수 데이터를 보여준다. 또한 사용자가 특정 영화의

My page 페이지에서 (회원가입과 로그인을 완료한) 사용자가 회원가입 시 입력한 사용자의 ID, Nickname, 선호장르, 나이 등의 개인정보를 확인할 수 있으며 선호 장르를 수정할 수 있다. 또한 사용자가 '좋아요'한 영화의 목록을 확인할 수 있으며 이를 삭제하거나 추가할 수 있다.

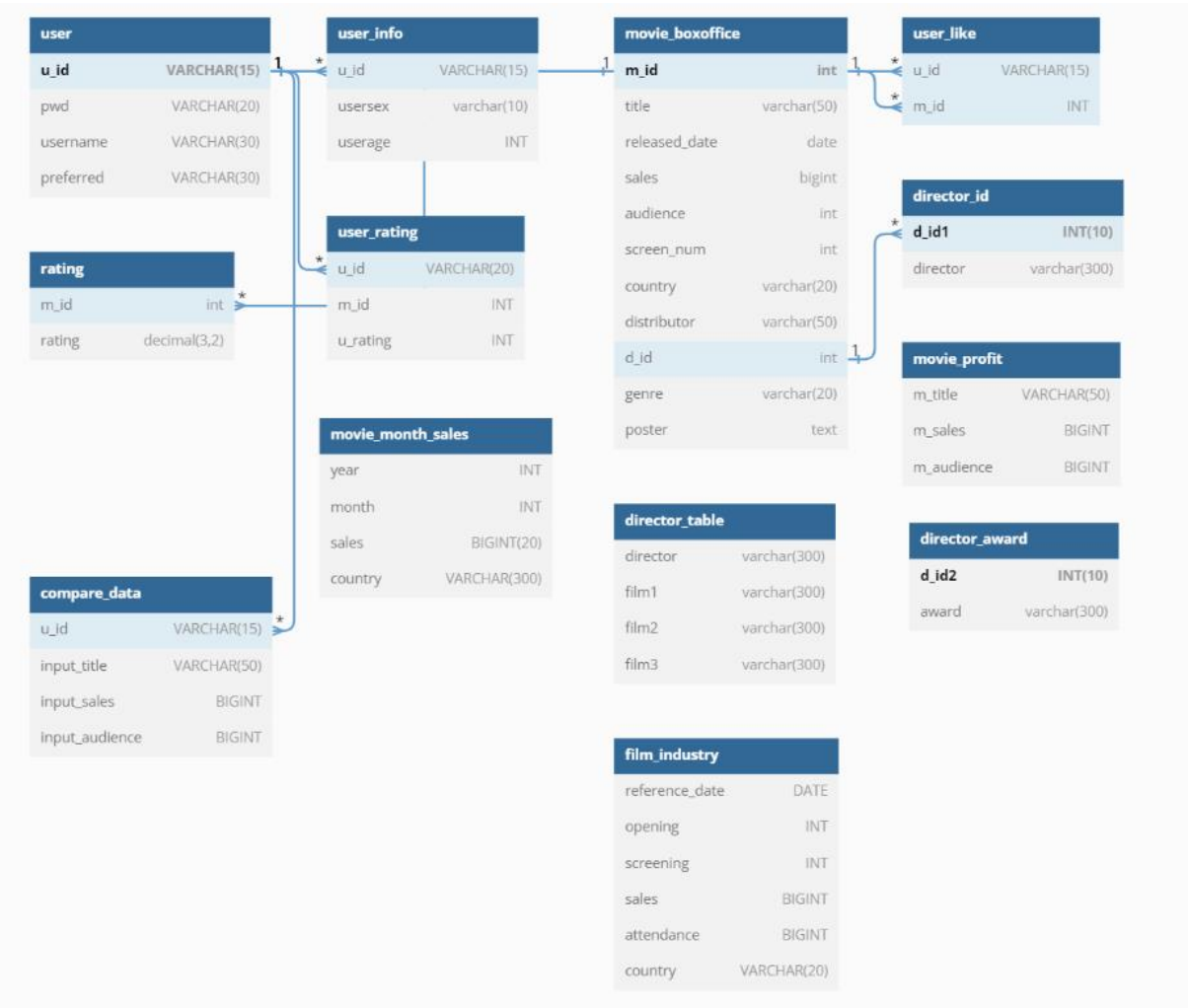
Login 페이지에서 아이디, 비밀번호를 입력하면 로그인을 할 수 있으며 Sign up 버튼을 누르면 Sign up 페이지로 넘어간다. Sign up 페이지에서 아이디, 비밀번호, 성별, 나이, 선호 장르를 입력하면 회원가입이 완료된다.

3. Schedule

날짜	일정	비고
9/29	팀 구성	팀 이름 선정, 자기 소개, 주제 논의
10/15	비대면 회의	주제 선정(기후) 및 proposal 보고서 작성
11/2	대면 회의	주제 변경(영화) 및 페이지 구성 논의
11/3	대면 회의	역할분담 및 앞으로의 일정 수립
11/4	개인 과제	데이터 수집 후 회의 -> 역대 박스오피스 100위 데이터로 결정
11/5	비대면 회의	xampp 환경 설정 및 데이터베이스 스키마 작성
11/6	비대면 회의	요구사항 분담 및 데이터 분석 방식 결정

11/7	대면 회의	백엔드 구현
11/10-11	대면 회의	sql 쿼리, 백엔드 코드 오류 사항 공유 후 개선
11/12-13	개인 과제	백엔드 구현 및 모든 요구사항 구현 여부 확인
11/14	대면 회의	팀원 별 페이지 연결 후 오류 확인
11/15	대면 회의	프론트엔드, 페이지 별 헤더 및 사이트 전체적인 스타일 통일
11/16	대면 회의	보고서 작성, 오류 수정, 설치환경 점검
11/17	대면 회의	보고서 작성, 오류 수정, 제출할 데이터베이스 스크립트 확인

4. DB schema as ER diagram



5. php structure

main.php

- **keywordSearch.php**
  - **datail2.php**
    - ratingInput.php
      - ratingInput_update.php
        - ratingInput_update_form.php
      - ratingInput_delete.php
- **director.php**
  - modifydirector.php
  - newdirector.php
  - delete.php
- **filter.php**
- **genre.php**
  - genre_response.php
- **dash.php**
- **sales_month.php**
  - sales_month_response.php
  - industry.php
    - compare_result.php
    - compare_modify.php
      - compare_modify_result.php
    - compare_delete.php
- **mypage.php**
  - mypage_genre_update.php
    - mypage_genre_update_response.php
  - mypage_like_insert.php
    - mypage_like_insert_genre.php
    - mypage_like_insert_genre_search.php
  - mypage_like_delete.php
    - mypage_like_delete_response.php
- **login.php**
  - login_result.php
  - logout.php
  - signup.php
    - signup_result.php

6. Requirements

(6-1) Theme of the project : Analysis of Movie Data

(6-2) SQL scripts

i. Creation

① Primary keys

-- 김다희

```
CREATE TABLE director_id(
    d_id1 INT(10) NOT NULL AUTO_INCREMENT,
    director varchar(300),
    PRIMARY KEY (`d_id1`)
);
```

```
CREATE TABLE director_award(
    d_id2 INT(10) NOT NULL AUTO_INCREMENT,
    award varchar(300),
    PRIMARY KEY (`d_id2`)
);
```

// 1971058 홍진서

```
// create.sql
CREATE TABLE user(
    u_id VARCHAR(15) NOT NULL PRIMARY KEY,
    pwd VARCHAR(20) NOT NULL,
    username VARCHAR(30) NOT NULL,
    preferred VARCHAR(30) NOT NULL
);
```

② Foreign keys

// 1971058 홍진서

```
// create.sql
CREATE TABLE compare_data(
    u_id VARCHAR(15) NOT NULL,
    input_title VARCHAR(50) NOT NULL,
    input_sales BIGINT NOT NULL,
    input_audience BIGINT NOT NULL,
```

```
    FOREIGN KEY (u_id) REFERENCES user(u_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

// 이유림

```
CREATE TABLE user_like(
    u_id VARCHAR(15),
    m_id INT,
    FOREIGN KEY (u_id) REFERENCES user(u_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (m_id) REFERENCES movie_boxoffice(m_id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE user_info(
    u_id VARCHAR(15) NOT NULL,
    usersex varchar(10) NOT NULL,
    userage INT NOT NULL,
    FOREIGN KEY (u_id) REFERENCES user(u_id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

// 조유담

```
-- 조유담3
CREATE TABLE user_rating(
    u_id VARCHAR(20) NOT NULL,
    m_id INT NOT NULL,
    u_rating INT NOT NULL,
    FOREIGN KEY (u_id) REFERENCES user(u_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
-- 조유담2
CREATE TABLE rating(
    m_id int NOT NULL,
    rating decimal(3, 2) NOT NULL,
    FOREIGN KEY (m_id) REFERENCES movie_boxoffice(m_id)
);
```

③ Index creation statement

// 홍진서

```
// create.sql
CREATE TABLE compare_data(
    u_id VARCHAR(15) NOT NULL,
    input_title VARCHAR(50) NOT NULL,
    input_sales BIGINT NOT NULL,
    input_audience BIGINT NOT NULL,
```

```
    FOREIGN KEY (u_id) REFERENCES user(u_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

// 김다희

```
CREATE TABLE director_table (
    director varchar(300),
    film1 varchar(300),
    film2 varchar(300),
    film3 varchar(300)

);
▷ Execute
CREATE INDEX index_dir ON director_table(film1,film2);
```

ii. Insert

```
INSERT INTO user (u_id, pwd, username, preferred) VALUES ('team10', 'team10',
'김이화', 'action');
```

iii. Drop

```
DROP TABLE team10.user;
```

iv. Dump

(6-3) INSERT, DELETE, UPDATE, SELECT related functions with dynamic queies

i. 조유담

① Insert : 사용자 평점 등록하기

```
$movie_id = mysqli_real_escape_string($conn, $_POST['movie_id']);

$movie_title = mysqli_real_escape_string($conn, $_POST['movie_title']);

$user_rating = mysqli_real_escape_string($conn, $_POST['num']);


$sql = "INSERT INTO user_rating VALUES ('$user_id', '$movie_id', '$user_rating')";
```

② Delete: 사용자 평점 삭제하기

```php
$movie_id = mysqli_real_escape_string($conn, $_POST['movie_id']);

$movie_title = mysqli_real_escape_string($conn, $_POST['movie_title']);

$user_rating = mysqli_real_escape_string($conn, $_POST['num']);


$sql = "DELETE FROM user_rating WHERE u_id = '$user_id' AND m_id = '$movie_id'";
```

③ Update: 사용자 평점 수정하기

```php
$movie_id = mysqli_real_escape_string($conn, $_POST['movie_id']);

$movie_title = mysqli_real_escape_string($conn, $_POST['movie_title']);

$user_rating = mysqli_real_escape_string($conn, $_POST['num']);


$sql = "UPDATE user_rating SET u_rating = '$user_rating' WHERE u_id = '$user_id' AND m_id = '$movie_id'";
```

④ Select: 사용자가 클릭한 영화의 상세 내용 보여주기

```php
session_start();
$user_id = $_SESSION['id'];
// $user = '도레미';

$conn = mysqli_connect("localhost", "team10", "team10", "team10");

$title = $_GET['m_title'];
$sql = "SELECT * FROM movie_boxoffice WHERE title = '$title'";
$result = mysqli_query($conn, $sql);
$row = mysqli_fetch_array($result);


$id = $row['m_id'];

$sql_d = "SELECT d_id FROM movie_boxoffice WHERE m_id = '$id'";
$result_d = mysqli_query($conn, $sql_d);
$row_d = mysqli_fetch_array($result_d);
$d_id= $row_d['d_id'];


$sql_director_name = "SELECT * FROM director_id WHERE d_id = '$d_id'";
$result_director_name = mysqli_query($conn, $sql_director_name);
$row_director_name = mysqli_fetch_array($result_director_name);

$sql_r = "SELECT * FROM rating WHERE m_id = '$id'";
$result_r = mysqli_query($conn, $sql_r);
$row_r = mysqli_fetch_array($result_r);


$sql_user_rating = "SELECT * FROM user_rating WHERE u_id = '$user_id' AND m_id = '$id'";
$result_user_rating = mysqli_query($conn, $sql_user_rating);
$row_user_rating = mysqli_fetch_array($result_user_rating);
```

ii. 홍진서

① Insert

```
// compare_result.php
$sql_insert = "INSERT INTO compare_data(u_id, input_title, input_sales,
input_audience) VALUES (?, ?, ?, ?)";
```

② Delete

```
// compare_delete.php
$sql_delete = "DELETE FROM compare_data WHERE u_id = ?";
```

③ Update

```
$sql_update = "UPDATE compare_data SET input_sales = ?, input_audience = ?
WHERE u_id = ?"; // compare_modify_result.php
```

④ Select with dynamic queies

```
//idustry.php
$sql_select = "SELECT * FROM compare_data WHERE u_id = ?";

// $sql_insert 실행 코드 (dynamic queries)
$user_id = $_SESSION['id'];
$input_title = $_POST['input_title'];
$input_sales = $_POST['input_sales'];
$input_audience = $_POST['input_audience'];

$stmt = $mysqli->prepare($sql_insert);
$stmt->bind_param("ssii", $user_id, $input_title, $input_sales,
                                                    $input_audience);
$stmt->execute();
$result = $stmt->get_result();
```

iii. 김다희

① Insert : 감독 정보(이름, 대표작3개, 수상내역)을 dynamic query로 여러개의 input을 받아서 데이터베이스에 insert함.

```
$sql="INSERT INTO  director_table (director, film1, film2, film3)
VALUES('{$_POST['director']}','{$_POST['film1']}','{$_POST['film2']}','{$_POST['film3']}')";
$sql_1="INSERT into director_award(award) VALUES('{$_POST['award']}')";
$sql_2="INSERT into director_id(director)
VALUES('{$_POST['director']}')";
```

② Delete : 입력창에 감독의 이름을 입력하면 해당되는 감독의 행이 테이블(director_table의 모든항목, director_award의 모든 항목)이 삭제됨.

```
$sql="DELETE FROM dt, da ,did using director_table as dt
    join director_id as did
    on dt.director=did.director
    join director_award as da
    on did.d_id1=da.d_id2
    WHERE dt.director='{$_GET['di']}'";
```

③ Update : 입력창에 감독의 이름을 검색하고, 수정 내역에서 dynamic query로 film1, film2, film3, award로 여러개의 입력을 입력받아서 감독의 내용을 update한다.

```
$sql_1="UPDATE director_table SET
director='{$_GET['director']}',
film1='{$_GET['film1']}',
film2='{$_GET['film2']}',
film3='{$_GET['film3']}'
WHERE director = '{$_GET['director']}'";

$sql_2="UPDATE director_award AS da
INNER JOIN director_id AS did
ON did.d_id1=da.d_id2
SET award='{$_GET['award']}'
WHERE director= '{$_GET['director']}'";
```

④ Select : 입력창에 감독의 이름을 검색하면, 해당되는 감독의 행의 테이블(director_table.director, director_table.film1, director_table.film2, director_table.film3, director_award.award)를 출력한다.

```
$sql ="SELECT di.d_id1, dt.director, dt.film1, dt.film2, dt.film3, da.award FROM
director_table AS dt
join director_id as di on dt.director=di.director
join director_award as da on di.d_id1=da.d_id2 WHERE dt.director='$director'";//%$director%
```

iv. 이유림

① Insert : mypage의 좋아요 목록에서 ADD 버튼을 클릭하면 원하는 장르의 영화 목록이 보여진다. 그 중, 좋아요 목록에 추가하고 싶은 영화를 클릭하면 user_like 테이블에 유저의 id와 movie의 id가 insert되어 mypage에서 확인할 수 있음

```php
$insert_query = "INSERT INTO user_like(u_id, m_id) VALUES (?, ?)";
$stmt = $mysqli->prepare($insert_query);
$stmt->bind_param("si", $user_id, $movie_id);
$insert_res = $stmt->execute();
```

② Delete : mypage 좋아요 목록에서 삭제할 영화를 선택하면 user_like 테이블에서 유저 id와 좋아요 목록에 추가되어 있던 movie id가 삭제됨

```php
$insert_query = "DELETE FROM user_like
                 WHERE m_id='$movie_id' AND u_id='$user_id'";
```

③ Update : mypage 정보 화면에서 edit preferred genre를 클릭하면 사용자의 선호 장르가 변경되어 user table의 preferred 항목이 update됨

```php
$genre_update = "UPDATE user SET preferred='" . $_POST['genre'] . "'
                 WHERE user.u_id= '" . $_SESSION['id'] . "'";
```

④ Select

```php
$sql = "SELECT poster,title, released_date, genre
    FROM movie_boxoffice WHERE movie_boxoffice.genre = '$post_genre'";
```

원하는 genre를 선택하면 poster, title, 개봉일, 장르 등을 select 하여 목록에 나타냄

```php
$nickname = $_SESSION['name'];
$mysqli = mysqli_connect("localhost", "team10", "team10", "team10");
$sql = "SELECT poster, title
        FROM movie_boxoffice
        LEFT JOIN user_like ON movie_boxoffice.m_id=user_like.m_id
        WHERE u_id = (SELECT user.u_id
                      FROM user
                      WHERE user.username='$nickname')";
```

외래키로 설정된 user_like 테이블의 u_id와 m_id를 이용하여 user_like에 등록되어 있는 영화의 제목, 포스터를 SELECT하여 마이페이지에 나타냄

(6-4) SELECT문

```php
// 1971058 홍진서
// industry.php
$sql_y = "SELECT year(reference_date) AS 'Year', country,
                 round(sum(attendance)) AS sum_attendance,
                 round(avg(attendance)) AS avg_attendance
          FROM film_industry
```

```
            WHERE year(reference_date) > 1000
            GROUP BY year(reference_date), country WITH ROLLUP";
```

(6-5) 3 kinds of advanced analysis functions

i. Group by + aggregation operations(SUM, AVG)

// 홍진서

```
// industry.php
$sql_y = "SELECT year(reference_date) AS 'Year', country,
                    round(sum(attendance)) AS sum_attendance,
                    round(avg(attendance)) AS avg_attendance
            FROM film_industry
            WHERE year(reference_date) > 1000
            GROUP BY year(reference_date), country WITH ROLLUP";
```

// 이유림: 매출 분석 페이지에서 년도를 기준으로 그룹화하여 한국 영화 산업에서의 분기별 매출 평균을 구함.

```
$sql = "SELECT movie_month_sales.year,month,format(avg(sales),0)
        FROM movie_month_sales
        where month BETWEEN 1 AND 3
        GROUP BY movie_month_sales.year,country
        Having country='한국'";
```

ii. Advanced analysis functions

① aggregates (sum, average, max, min, etc) + grouping

// 홍진서

```
// industry.php
$sql_m = "SELECT month(reference_date) AS 'Month', country,
                    round(sum(attendance)) AS sum_attendance,
                    round(avg(attendance)) AS avg_attendance
            FROM film_industry
            WHERE year(reference_date) = ?
            GROUP BY month(reference_date), country WITH ROLLUP";
```

//이유림

```
$sql = "SELECT movie_month_sales.year,country,format(sum(sales),0)
        FROM movie_month_sales
        GROUP BY movie_month_sales.year,country
        ORDER BY movie_month_sales.year";
```

② roll up

// 홍진서

```
// industry.php
$sql_m = "SELECT month(reference_date) AS 'Month', country,
                 round(sum(attendance)) AS sum_attendance,
                 round(avg(attendance)) AS avg_attendance
          FROM film_industry
          WHERE year(reference_date) = ?
          GROUP BY month(reference_date), country WITH ROLLUP";
```

③ ranking

// 조유담

// 평점 탑10 영화

```
$sql_join = "SELECT movie_boxoffice.m_id, title, rating,
rank() over(order by rating desc) AS ranking
FROM movie_boxoffice LEFT JOIN rating ON movie_boxoffice.m_id = rating.m_id" ;

$result_join = mysqli_query($conn, $sql_join);



$top_10 = 9;
for($i = 0; $i <= $top_10; $i++){
  $row_join = mysqli_fetch_array($result_join);
?>
```

// 매출 탑10위 영화 (한국 및 외국영화)

```
$sql_join = "SELECT *,
rank() over(order by sales desc) AS ranking
FROM movie_boxoffice";

$result_join = mysqli_query($conn, $sql_join);

$top_10 = 9;
for($i = 0; $i <= $top_10; $i++){
    $row_join = mysqli_fetch_array($result_join);
?>
```

// 관람객 수 탑 10위 영화

```
$sql_join = "SELECT *,
rank() over(order by audience desc) AS ranking
FROM movie_boxoffice
ORDER BY ranking LIMIT 10";
```

// 매출 탑 10위 한국 영화

```
$sql_join = "SELECT *,
rank() over(partition by country order by sales desc) AS ranking
FROM movie_boxoffice
ORDER BY country,ranking LIMIT 10";
```

(6-6) User input with various types

// 홍진서

```
<!—1971058 홍진서 -->
<!-- filter.php -->
<form method="POST">
    <select name="country">
        <option value="Korea">Korea</option>
        <option value="USA">Overseas</option>
    </select>

    <input type='radio' name='rate' value='5'checked />over 5
    <input type='radio' name='rate' value='6' />over 6

    over <input type="text" name="AudTxtMin" value="0"> million,
    under <input type="text" name="AudTxtMax" value="20"> million

    <input type='radio' name='year' value='0' />before 2005
    <input type='radio' name='year' value='2005' />2005 to 2009
</form>

// form에서 입력받은
$sql = "SELECT * FROM movie_boxoffice
          INNER JOIN rating ON movie_boxoffice.m_id = rating.m_id
        WHERE country = '$country' AND rating.rating >= $rate
                  AND (audience BETWEEN $audMin AND $audMax)
                  AND (YEAR(released_date) BETWEEN $year AND $year+4)";
```

// 김다희

```
$sql="INSERT INTO  director_table (director, film1, film2, film3)
VALUES('{$_POST['director']}','{$_POST['film1']}','{$_POST['film2']}','{$_POST['film3']}')";
$sql_1="INSERT into director_award(award) VALUES('{$_POST['award']}')";
$sql_2="INSERT into director_id(director)
VALUES('{$_POST['director']}')";
```

```
$sql_1="UPDATE director_table SET
director='{$_GET['director']}',
film1='{$_GET['film1']}',
film2='{$_GET['film2']}',
film3='{$_GET['film3']}'
WHERE director = '{$_GET['director']}'";

$sql_2="UPDATE director_award AS da
INNER JOIN director_id AS did
ON did.d_id1=da.d_id2
SET award='{$_GET['award']}'
WHERE director= '{$_GET['director']}'";
```

(6-7) PHP sessions

//홍진서

```
// 1971058 홍진서
// login_result.php
if (mysqli_num_rows($result)) {
    $row = $result->fetch_array(MYSQLI_ASSOC);
    $_SESSION['id'] = $row['u_id'];
    $_SESSION['name'] = $row['username'];
}

<!--main.php-->
<header id="main_header">
    <ul class="header_ul">
        <?php
        if (isset($_SESSION['name'])) { ?>
            <li><a href="./logout.php"> Log out</a></li>
        <?php
        } else { ?>
            <li><a href="./login.php"> Login</a></li>
        <?php
        }
        ?>
    </ul>
</header>

// logout.php
```

```php
session_start();
session_destroy();
echo "<script>location.href='main.php';</script>";
```

//김다희

```php
<div class="backgroundImage">

    <div class="mainContainer">
        <div>
            <?php if (isset($_SESSION['name'])) { ?>
                <h1>Welcome, <?php echo $_SESSION['name']?></h1>
            <?php
            } else { ?>
                <h1>EXPLORE MOVIE!</h1>
                <h2>Please log in</h2>
            <?php
            }
            ?>
            <?php
                if (isset($_SESSION['name'])) { ?>
                <form action="keywordSearch.php" method="get">
                <input id="searchdata" type="textbox" name="keyword" placeholder="영화 제목을 검색하세요" >

                <button type="submit" value="Search">검색</button>
            </form>

            <?php
                } else {
                    ?>
                <div class="center-button">
                <Button onclick="location.href='login.php'"> LOGIN</Button></div>
                <?php
                }
                ?>
            </form>
            </div>
            <div>
        <p> Let's see which movie is interesting</p>
    </div>
</div>
```

(6-8) Transaction

// 이유림

마이페이지에서 사용자의 선호 장르를 변경할 때, user 테이블에서 user id와 선호 영화 장르를 가져오는 SELECT문과 선호 영화 장르를 변경하는 UPDATE문을 하나의 트랜잭션으로 묶고, 둘 중 하나의 쿼리에서 에러가 나면 rollback이 되도록 처리했다.

```php
mysqli_begin_transaction($mysqli);

try {
    //$sql1 = "SELECT id, preferred FROM user WHERE user.username = '$_SESSION['name']";
    $user_info = "SELECT u_id, preferred FROM user WHERE user.username = '" . $_SESSION['name'] . "'";
    $row = mysqli_fetch_array(mysqli_query($mysqli, $user_info));
    $id = $row[0];
    $genre = $row[1];

    $genre_update = "UPDATE user SET preferred='" . $_POST['genre'] . "'
                    WHERE user.u_id= '" . $_SESSION['id'] . "'";
    $res = mysqli_query($mysqli, $genre_update);

    mysqli_commit($mysqli);
    echo 'commit';
} catch (mysqli_sql_exception $exception) {
    mysqli_rollback($mysqli);
    echo 'rollback';
}

mysqli_commit($mysqli);
```

7. Team member responsibilities

- 조유담 : 백엔드, 프론트엔드, 데이터베이스
- 홍진서 : 백엔드, 프론트엔드, 데이터베이스
- 김다희 : 백엔드, 프론트엔드, 데이터베이스
- 이유림 : 백엔드, 프론트엔드, 데이터베이스

8. Detail implementation results by each team member

(1) 조유담

1. Detail 페이지

A. 페이지 왼쪽에 영화 포스터를 보여주고, 오른쪽에 영화의 상세 정보를 보여준다. 오른쪽 하단에는 평점을 입력, 수정, 삭제 폼이나 버튼이 존재함.

```php
$sql = "SELECT * FROM movie_boxoffice WHERE title = '$title'";
```

B. get 방식으로 이전 페이지에서 받은 '영화 제목'을 Where문의 조건으로 사용해서 movie_boxoffice 테이블에서 영화의 제목, 장르, 국적, 개봉일, 매출액, 관객수, 스크린 수, 배급사, 평점을 사용자에게 보여줌. 위의 쿼리를 이용해 movie_boxoffice 테이블에서 영화 아이디와 감독 아이디를 가져옴.

```php
$sql_r = "SELECT * FROM rating WHERE m_id = '$id'";
```

C. 위의 쿼리에서 가져온 영화 아이디를 이용해 rating 테이블에서 영화의 평점을 가져와 사용자에게 보여줌.

```
$sql_d = "SELECT d_id FROM movie_boxoffice WHERE m_id = '$id'";
$sql_director_name = "SELECT * FROM director_id WHERE d_id = '$d_id'"
```

D. 위에서 가져온 감독 아이디를 이용해서 director_id 테이블에서 감독 이름을 찾아서 사용자에게 보여줌.

```
$sql_user_rating = "SELECT * FROM user_rating
WHERE u_id = '$user_id' AND m_id = '$id'";
```

E. 또한 사용자가 해당 영화에 대한 평점을 이미 등록했다면, 세션을 통해 받은 사용자의 아이디와 앞서 가져온 영화 아이디를 이용해 user_rating 테이블에서 사전에 등록한 평점을 사용자에게 보여줌. 만약 사용자가 자신의 평점을 아직 등록하지 않았다면, 평점을 등록하는 창이 보임. select 태그를 통해 사용자가 자신의 평점을 등록하면 post 방식으로 ratingInput.php 파일에 영화 아이디와 제목과 사용자의 평점이 넘어감. ratingInput.php에서는 detail2.php에서 전달 받은 값들을 이용해 user_rating 테이블에 유저 id, 영화 id, 유저가 등록한 평점이 삽입됨

```
$sql = "INSERT INTO user_rating VALUES ('$user_id', '$movie_id', '$user_rating')";
```

또한 사용자가 자신이 입력한 평점을 수정하고자 한다면, update my rating 버튼을 누르면 내가 등록한 평점을 수정하는 ratingInput_update_form.php 페이지로 넘어감. 처음에 평점을 등록할 때와 같은 폼에 수정된 평점을 입력하고 등록을 누르면 ratingInput_update.php에서 평점 수정 쿼리를 실행하고 다시 detail.2.php에서 수정된 평점을 보여줌.

```
$sql = "UPDATE user_rating SET u_rating = '$user_rating' WHERE u_id = '$user_id' AND m_id = '$movie_id'";
```

F. 또한 사용자가 자신이 입력한 평점을 삭제하고자한다면, delete my rating 버튼을 누르면 ratingInput_delete.php 파일에서 내가 등록한 평점을 삭제하는 쿼리를 실행하고 detail.2.php에서 평점 등록하기 폼을 보여줌.

2. Dashboard (dash.php)

A. '평점 탑 10위 영화'와 '매출 탑 10위 영화(한국, 외국영화 모두 포함)'를 '테이블'로 '관람객 수 탑 10위 영화' 와 ' 매출 탑 10위 한국 영화'를 bar chart로 보여줌

B. '평점 탑 10위 영화' - rating 테이블과 movie_boxoffice 테이블을 조인해서 rank 함수를 이용해 평점을 기준으로 내림차순으로 정렬함. 이후 위에서부터 10개의 영화를 출력해 사용자에게 보여줌.

C. '매출 탑 10위 영화(한국, 외국영화 모두 포함)' - movie_boxoffice 테이블에서 rank함수를 이용해 매출을 기준으로 내림차순 정렬함. 이후 위에서부터 10개의 영화를 출력해 사용자에게 보여줌.

D. '관람객 수 탑 10위 영화' - movie_boxoffice 테이블에서 rank함수를 이용해 관람객 수를 기준으로 내림차순 정렬함. 쿼리에 Limit 10을 붙여 탑 10위의 영화를 사용자에게 보여줌.

E. '매출 탑 10위 한국 영화' - movie_boxoffice 테이블에서 rank 함수를 이용해 국적으로 partition by 적용해 국적과 매출을 기준으로 내림차순 정렬함. 쿼리에 Limit 10을 붙이고 한국 partition에서 탑 10위의 영화를 사용자에게 보여줌.

(2) 홍진서

1. 로그인

A. PHP 및 query: 사용자가 HTML form에 입력한 id와 password를 POST 방식으로 받아와 Prepared Statement를 사용해 DB의 user table에 입력값과 일치하는 행의 모든 값을 가져오는 쿼리를 작성했다. 입력값이 table에 있을 경우, 사용자의 id와 name을 세션에 저장하고 main.php페이지로 넘어간다. 입력값이 table에 없을 경우, 로그인에 실패했다는 알림을 보내고 login.php 페이지로 돌아간다.

2. 회원가입

A. PHP 및 query: 사용자로부터 ID, 비밀번호, 이름을 직접 입력받고 성별, 연령대, 선호 영화 장르를 radio button으로 입력받는다. 입력받은 ID가 이미 user table에 저장되어 있는지 확인하는 쿼리를 작성했다. 이미 존재하는 ID라면 다시 회원가입 창으로 돌아가고 존재하지 않는 ID라면 입력받은 ID/비밀번호/이름/선호장르를 user table에, ID/성별/연령대를 user_info table에 저장한다.

3. 영화 제목 및 감독명 검색

A. PHP 및 query: left join된 movie_boxoffice table과 director_id table의 키 title 혹은 director에 사용자가 입력한 keyword가 포함된다면, 해당 행을 가져오는 SELECT문 쿼리를 작성했다. 조건에 맞는 모든 행의 영화 제목, 개봉일자, 관객 수, 감독명, 장르를 테이블 형식으로 화면에 출력하고, 영화 제목을 클릭하면 해당 영화의 detail2.php 페이지로 이동한다.

4. 필터링

A. PHP 및 query: inner join된 movie_boxoffice table과 rating table에서 사용자가 입력한 조건(영화의 제작 국가, 평가, 관객 수, 개봉 년도)에 맞는 행을 가져오는 쿼리를 작성했다. 제작 국가의 경우 정확히 일치하는 값을 찾고, 평가와 관객 수는 입력값보다 큰 값을 찾고, 개봉 년도는 'between 개봉년도 and 개봉년도+4'에 해당하는 값을

찾는다.

```
$sql = "SELECT * FROM movie_boxoffice
                        INNER JOIN rating
                        ON movie_boxoffice.m_id = rating.m_id
                    WHERE country = '$country' AND rating.rating >= $rate
AND (audience BETWEEN $audMin AND $audMax) AND (YEAR(released_date)
BETWEEN $year AND $year+4)";
```

B. Front-end: '제작 국가'는 HTML의 select-option tag를 통해 국내 또는 해외 선택지를 제공했다. '평가'는 radio button으로 숫자를 선택할 수 있으며, 기타를 선택하면 직접 입력이 가능하다. '관객 수'는 radio button으로 숫자를 선택할 수 있으며, 기타를 선택하면 직접 범위 지정이 가능하다. '개봉 년도'는 radio button으로 값을 선택한다.

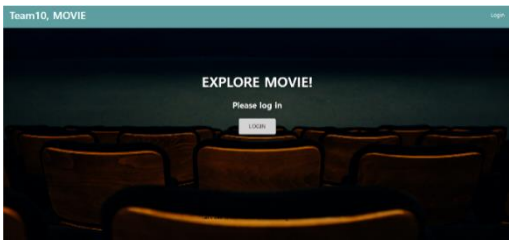
5. 영화 순위 및 영화 관객수 통계

A. PHP 및 query: 사용자의 입력값(영화제목, 매출, 관객수)을 한국 역대 박스오피스 정보와 비교한다. 입력받은 매출값과 관객수를 각각의 쿼리에 넣어 PERCENT_RANK() 함수를 통해 입력한 영화가 상위 몇 퍼센트에 해당하는지 알아낸다. 해당 값은 수정, 삭제, 재입력이 가능하다.

(3) 김다희

1. 메인화면

메인 화면에서 session을 이용해서 로그인이 되어있지 않은 상태에서는 홈페이지를 이용할 수 없다. 처음 메인화면에서 login버튼을 눌러서 signup을 하면 회원가입을 할 수 있다. 회원가입이후 로그인을 진행하면, session을 이용하여 웹 사이트를 이용할 수 있는 메인화면이 나온다. 헤더의 각각의 버튼을 클릭하여 해당하는 항목을 검색할 수 있다. 또한 메인화면의 검색창에서 감독이름이나, 영화제목을 입력하면, 해당 키워드에 맞는 항목을 자세히 검색할 수 있다



```php
<div class="backgroundImage">

    <div class="mainContainer">
        <div>
            <?php if (isset($_SESSION['name'])) { ?>
                <h1>Welcome, <?php echo $_SESSION['name']?></h1>
            <?php
            } else { ?>
                <h1>EXPLORE MOVIE!</h1>
                <h2>Please log in</h2>
            <?php
            }
            ?>
            <?php
                if (isset($_SESSION['name'])) { ?>
                <form action="keywordSearch.php" method="get">
                <input id="searchdata" type="textbox" name="keyword" placeholder="영화 제목을 검색하세요" >

                <button type="submit" value="Search">검색</button>
            </form>

            <?php
                } else {
                    ?>
                <div class="center-button">
                <Button onclick="location.href='login.php'"> LOGIN</Button></div>
                <?php
                }
                ?>
            </form>
            </div>
```

2. 감독 이름 검색(director)

director.php에서는 감독의 director_table의 director와 director_id의 director을 이용해 join하고, director_id의 d_id1(감독의 아이디)와 director_award의 d_id2를 이용해 join한다. 그리고, director을 input으로 받아서 WHERE문에서 director name을 검색할 때 사용한다. 검색 결과는 table형식으로 받으며, 항목은 Director name, Film1, Film2, Film3, Award를 이용해 보여준다.


```php
$sql ="SELECT di.d_id1, dt.director, dt.film1, dt.film2, dt.film3, da.award FROM
director_table AS dt
join director_id as di on dt.director=di.director
join director_award as da on di.d_id1=da.d_id2 WHERE dt.director='$director'";//%$director%
```

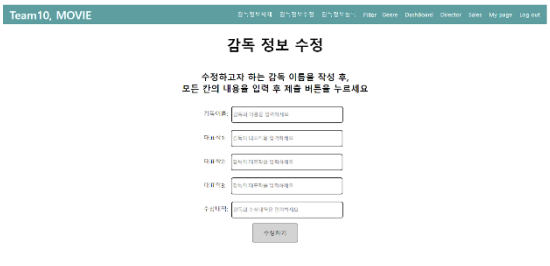
3. 감독 정보 등록 (newdirector)

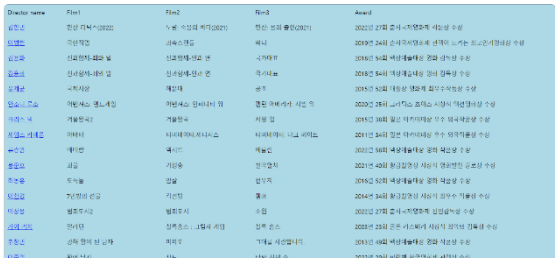
newdirector.php에서는 감독의 정보를 새롭게 등록할 수 있다. 제출버튼을 누르면 자동으로 테이블에 저장된다. POST를 사용하여 감독의 이름, 대표작3개, 수상내역을

입력받고, INSERT문을 사용해서 director_table, director_id, director_award 테이블에 각각 insert해준다.
각각의 테이블에서 d_id1와 d_id2는 Auto increment를 설정하여 새로운 항목을 입력받아도 각 테이블에서 같은 id number를 가질 수 있도록 설정한다.

4. 감독 정보 수정 (modifydirector)

modifydirector.php에서는 먼저 director_table과 director_award에 있는 모든 감독에 대한 정보를 테이블로 보여준다.
여기에서 감독의 이름을 입력하고 수정할 부분을 입력한 후, 제출버튼을 누르면 해당 감독의 정보를 UPDATE할 수 있다.





```
$sql_1="UPDATE director_table SET
director='{$_GET['director']}',
film1='{$_GET['film1']}',
film2='{$_GET['film2']}',
film3='{$_GET['film3']}'
WHERE director = '{$_GET['director']}'";

$sql_2="UPDATE director_award AS da
INNER JOIN director_id AS did
ON did.d_id1=da.d_id2
SET award='{$_GET['award']}'
WHERE director= '{$_GET['director']}'";
```

5. 감독 정보 삭제 (delete)

delete.php에서는 감독의 이름을 입력하면, director_table, director_id, director_award에서 해당 감독의 모든 정보를 삭제한다.

```
$sql="DELETE FROM dt, da ,did using director_table as dt
    join director_id as did
    on dt.director=did.director
    join director_award as da
    on did.d_id1=da.d_id2
    WHERE dt.director='{$_GET['di']}'";
```

6. **Dynamic queries**

   3의 감독정보 등록과, 4의 감독정보 수정부분에서 사용자에게 여러 항목을 입력 받아서 query문에 이용한다.

7. **Index**

   director_table의 award1, award2에 인덱싱 기법을 사용하여 인덱스를 만들었다.

(4) 이유림

1. **Mypage**

   A. User 테이블에서 user의 id, username, 선호 장르를 select 해오고, user_info 테이블에서 로그인 된 유저의 age를 가져온다. 'Edit preferred Genre' 버튼을 통해 mypage_like_delete_response.php로 이동하여 선호 장르를 편집할 수 있는데, 트랜잭션을 삽입하여 쿼리가 실패하면 rollback이 되도록 조치했다. 실행이 성공하면 선호 장르가 새롭게 업데이트 된 mypage로 돌아온다.

   B. 외래키로 설정된 user_like 테이블의 u_id와 m_id를 이용하여 각각 user과 movie 테이블로 JOIN하고, user_like에 등록되어 있는 영화의 제목과, 포스터를 SELECT하여 현재 로그인 된 유저의 마이페이지의 좋아요 목록에 나타낸다.

```php
$nickname = $_SESSION['name'];
$mysqli = mysqli_connect("localhost", "team10", "team10", "team10");
$sql = "SELECT poster, title
        FROM movie_boxoffice
        LEFT JOIN user_like ON movie_boxoffice.m_id=user_like.m_id
        WHERE u_id = (SELECT user.u_id
                      FROM user
                      WHERE user.username='$nickname')";
```

   C. 좋아요 목록 삭제 버튼을 누르면,mypage_like_delete_response.php로 이동하게 되고, 삭제할 영화를 선택하면 세션을 통해 가져온 user id와 삭제를 위해 선택된 영화의 이름을 이용하여 movie_boxoffice 테이블에서 movie id를 가져온 후 user_like 테이블에서 delete한다. 삭제 완료가 되면 자동으로 DELETE가 반영된 mypage로 돌아온다.

   D. 좋아요 목록 추가 버튼을 누르면 mypage_like_insert_genre_search 페이지로 이동하고 여기서 사용자의 편의를 위해 장르별 보기를 제공한다. 원하는 영화를 선택하면, user_like 테이블에 로그인 된 유저의 u_id와 선택한 영화의 m_id가 insert 된다. 그 후 자동으로 mypage에 돌아가면 추가한 영화가 함께 목록에 보여진다.

2. Genre

A. movie_boxoffice 테이블에 있는 전체 영화 목록을 가져와 장르별로 포스터와 영화 제목을 볼 수 있다. 사용자가 Radio button을 이용하여 원하는 장르를 선택하면, 해당 값을 POST 방식으로 넘겨 movie_boxoffice로부터 preferred column의 element와 넘긴 값(장르)이 같은 영화의 포스터와 제목을 선택하여 모두 출력한다

3. 영화 산업 매출

A. Sum of yearly sales를 선택하면 movie_month_sales 테이블에서 년도와 국가를 기준으로 GROUP BY 한 후, year, country, sum(sales)을 연도 순으로 정렬된 테이블 형태로 보여준다.

```
$sql = "SELECT movie_month_sales.year,country,format(sum(sales),0)
    FROM movie_month_sales
    GROUP BY movie_month_sales.year,country
    ORDER BY movie_month_sales.year";
$res = mysqli_query($mysqli, $sql);

while ($row = mysqli_fetch_array($res)) { ?>
    <table align="center" style="display: inline-block;">
        <thead>
            <tr>
                <th width=100>Year</th>
                <th width=100>country</th>
                <th width=200>sales</th>
            </tr>
        </thead>
        <tbody align=" center">
            <tr>
                <td><?php echo $row[0]; ?></td>
                <td><?php echo $row[1]; ?></td>
                <td><?php echo $row[2]; ?></td>
            </tr>
        </tbody>
    <?php } ?>
    </table>
```

B. Quarterly Sales Average를 선택하면 movie_month_sales 테이블에서 년도와 국가를 기준으로 GROUP BY 한 후, year, country, avg(sales)를 표 형태로 보여준다. Where 절에서 BETWEEN을 활용하여 각 년도 별 매출 평균을 분기별로 나누어 나타낸다.