Big Data Application (01)

# Team21 Lucky Movie: Analysis of Movie Data

**KIM KYURI,** ***1771113 and*** **Moon Jee Hyun,** ***1771093***

**SUMMARY** In the big data application class, team21 (lucky team) created a movie data analysis site using Kaggle's movie data.
***key words:*** *Data analysis, movie, php, MySQL*

## 1. Introduction

By using Kaggle's TMDB 5000 Movie Dataset, we implement a website that can search information related to movies. The total number of tables used is 6 and the total number of records is 224. When users log in to the site, they can search for basic movie-related information.

## 2. Requirements and Database Design

### 2.1 Requirements

Additional features of the site are as follows. 1st Displays the number of movies per year. 2nd When you search for an actor, the movies that the actor plays are displayed. 3rd When you search for a genre, movies of that genre are displayed. 4th It displays the ranking of movies by budget and displays (revenue/budget) * 100 indicator information. And finally, movie information is displayed by ranking in order of popularity.

### 2.2 ER Diagram for Database

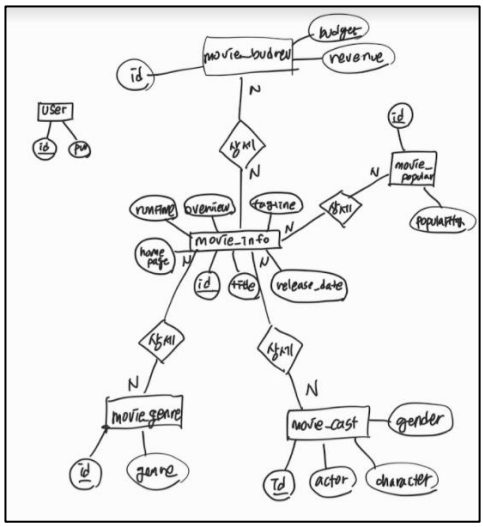


**Fig. 1** ER Diagram for Database

### 2.3 User Table

This is a table that stores user ID and password. The relationship between the user and the movie table can further develop the project by adding a personalized movie recommendation system in the future.

**Table 1** User

| | |
|---|---|
| **id** | varchar(10) |
| **pw** | varchar(10) |

### 2.4 Movie_info Table

This is a table that contains overall data such as the title, content, and screening date of the movie. PK is the ID of the movie. In addition, the PK has a foreign key relationship with other related tables. The user finds the title of the movie through the ID existing in the table through search. Detailed columns can be found in the ER diagram above.

**Table 2** Movie_info

| | |
|---|---|
| **id** | int(6) |
| **title** | varchar(43) |
| **overview** | varchar(638) |
| **tagline** | varchar(53) |
| **release_date** | varchar(10) |
| **runtime** | int(3) |
| **homepage** | varchar(82) |

### 2.5 Movie_genre Table

This is a table that contains data on the genre of a movie. PK

corresponds to id and is connected to Movie_info through the PK. After joining, the title of movie_info is displayed. The user can click the genre tab to know the ranking and title corresponding to the genre of the table.

**Table 3** Movie_genre

| | |
|---|---|
| id | int(5) |
| name | varchar(15) |
| movie_id | int(6) |
| movie_name | varchar(43) |

2.6 Movie_cast Table

This is a table that contains data on the casting of a movie. PK corresponds to id and is connected to Movie_info through the PK. After joining, the title of movie_info is displayed. The user can find the actor by searching for the actor he wants to search for, and accordingly he can know the title.

**Table 4** Movie_cast

| | |
|---|---|
| actor | varchar(20) |
| character | varchar(33) |
| gender | varchar(5) |
| movie_id | int(6) |
| movie_name | varchar(43) |

2.7 Movie_popular Table

This table contains data on the popularity of movies. PK corresponds to id and is connected to Movie_info through the PK. After joining, the title of movie_info is displayed. The user can click the genre tab to know the ranking and title by popularity corresponding to the genre of the table.

**Table 5** Movie_popular

| | |
|---|---|
| movie_id | int(6) |
| movie_name | varchar(43) |
| popularity | decimal(9,6) |

2.8 Movie_budrev Table

This is a table that contains data on the budget of the movie. PK corresponds to id and is connected to Movie_info through the PK. After joining, the title of movie_info is displayed. The user can click the budget tab to see the ranking, budget, and title for the movies in the table.

**Table 6** Movie_budrev

| | |
|---|---|
| movie_id | int(6) |
| movie_name | varchar(43) |
| budget | int(9) |
| revenue | bigint(10) |

## 3. System Design and Implementation

3.1 Diagram for PHP

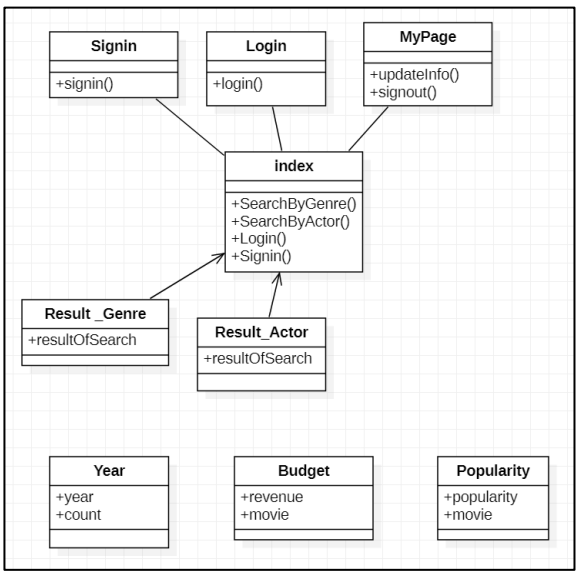


**Fig. 2** Diagram for PHP

3.1 Index

At the top of every page, including the index, there is a button that takes you to the login window. If the user is logged in, the Myhome page to edit his/her information or a button to logout is activated.

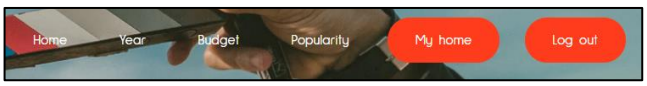


**Fig. 3** Implementation of Index

### 3.2 Signin

Users enter their personal information on the Signin page. At this time, input information in the form of a drop-down box as well as a text window. The password value and id value are saved through the insert received from the user. PHP-MySQL transaction was used.

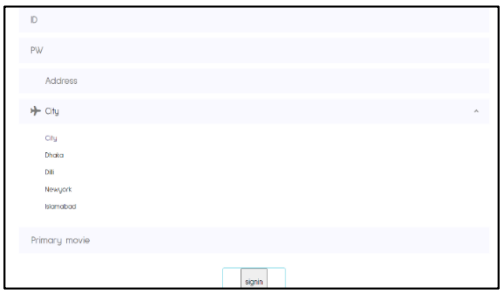


**Fig. 4** Implementation of Signin

### 3.3 Login

The user can log in through the Login page.

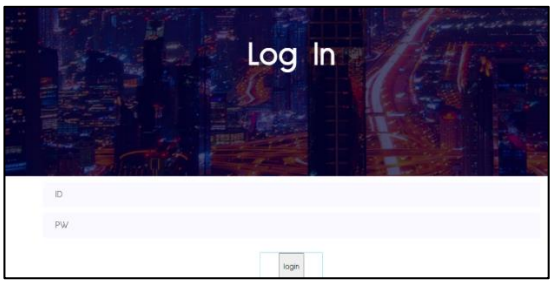


**Fig. 5** Implementation of Login

### 3.4 Mypage

Logged-in users can modify or withdraw their information through Mypage.


**Fig. 6** Implementation of Mypage

### 3.5 Result_genre

This is the first advanced analysis function. When a user searches for a genre, the movies appear in the movie_genre table. Also, the id value becomes FK, and the title of movie_info is displayed.


| index | title | genre |
|---|---|---|
| 20 | Avatar | Action |

**Fig. 7** Implementation of Result_genre

### 3.6 Result_actor

This is the second advanced analysis function. When a user searches for an actor's name, the actor's name is displayed in the movie_cast table. Also, the id value becomes FK, and the title of movie_info is displayed.


| index | title | character | gender |
|---|---|---|---|
| 1 | Pirates of the Caribbean: At World's End | Captain Jack Sparrow | Man |
| 2 | Pirates of the Caribbean: Dead Man's Chest | Captain Jack Sparrow | Man |
| 3 | The Lone Ranger | Tonto | Man |
| 4 | Pirates of the Caribbean: On Stranger Tides | Captain Jack Sparrow | Man |

**Fig. 8** Implementation of Result_actor

### 3.7 Year

This is the third advanced analysis function. Users can check the movies released by year through the year tab. Here, a query that includes a group by function by count and year was used.

| index | year | count |
|---|---|---|
| 11 | 2016 | 1 |
| 10 | 2015 | 2 |
| 9 | 2014 | 1 |
| 8 | 2013 | 2 |

**Fig. 9** Implementation of Year

### 3.8 Budget

This is the fourth advanced analysis function. Users can

figure out the budget of the movies. At this time, you can also check the contents of income through the income formula calculated by our team through revenue versus budget. The income formula is as follows.

$$income = (revenue/budget) * 100$$

| rank | title | budget | revenue | income |
|---|---|---|---|---|
| 1 | Avatar | 237000000 | 2787965087 | 1176.35657637932 |
| 2 | The Avengers | 220000000 | 1519557910 | 690.708140909091 |
| 3 | Avengers: Age of Ultron | 280000000 | 1405403694 | 501.929889071429 |

**Fig. 10** Implementation of Budget

3.9 Popularity

This is the fifth advanced analysis function. Ranking by popularity can be checked.

| rank | title | popularity |
|---|---|---|
| 1 | Batman v Superman: Dawn of Justice | 155.790452 |
| 2 | Avatar | 150.437577 |
| 3 | Pirates of the Caribbean: Dead Man's Chest | 145.847379 |
| 4 | The Avengers | 144.448633 |
| 5 | Pirates of the Caribbean: At World's End | 139.082615 |

**Fig. 11** Implementation of Popularity

## 4. Conclusions and Future Works

By using PHP and MySQL, we created a site where users can view movie data and check analysis results when users log in. The total number of tables created by refining the data is 6, and the total number of records is 224. There are 5 advanced analysis functions on the site, and various analysis will be possible by inserting a larger amount of records into the database. In addition, we plan to implement a movie favorite function and a personalized movie recommendation system as a customized service.