**Team member**: Yang Dai, Tao Li

**Enrolled in**: 415

Project domain

It’s a movie statistic database. This database is focus on the basic information of movies and artists involved in that movie and also some awards they won.

# Data Loading

Half of the data in our database is obtained from the API provide by IMDB. It take http request as input and return data as json type. URL is like [*http://mymovieapi.com/?title=tt123456&type=json&plot=simple&episode=1&lang=en-US&aka=simple&release=simple&business=1&tech=0*](http://mymovieapi.com/?title=tt123456&type=json&plot=simple&episode=1&lang=en-US&aka=simple&release=simple&business=1&tech=0)

The whole process to get data is implement by a python file named *getData.py* which is attached in the package.

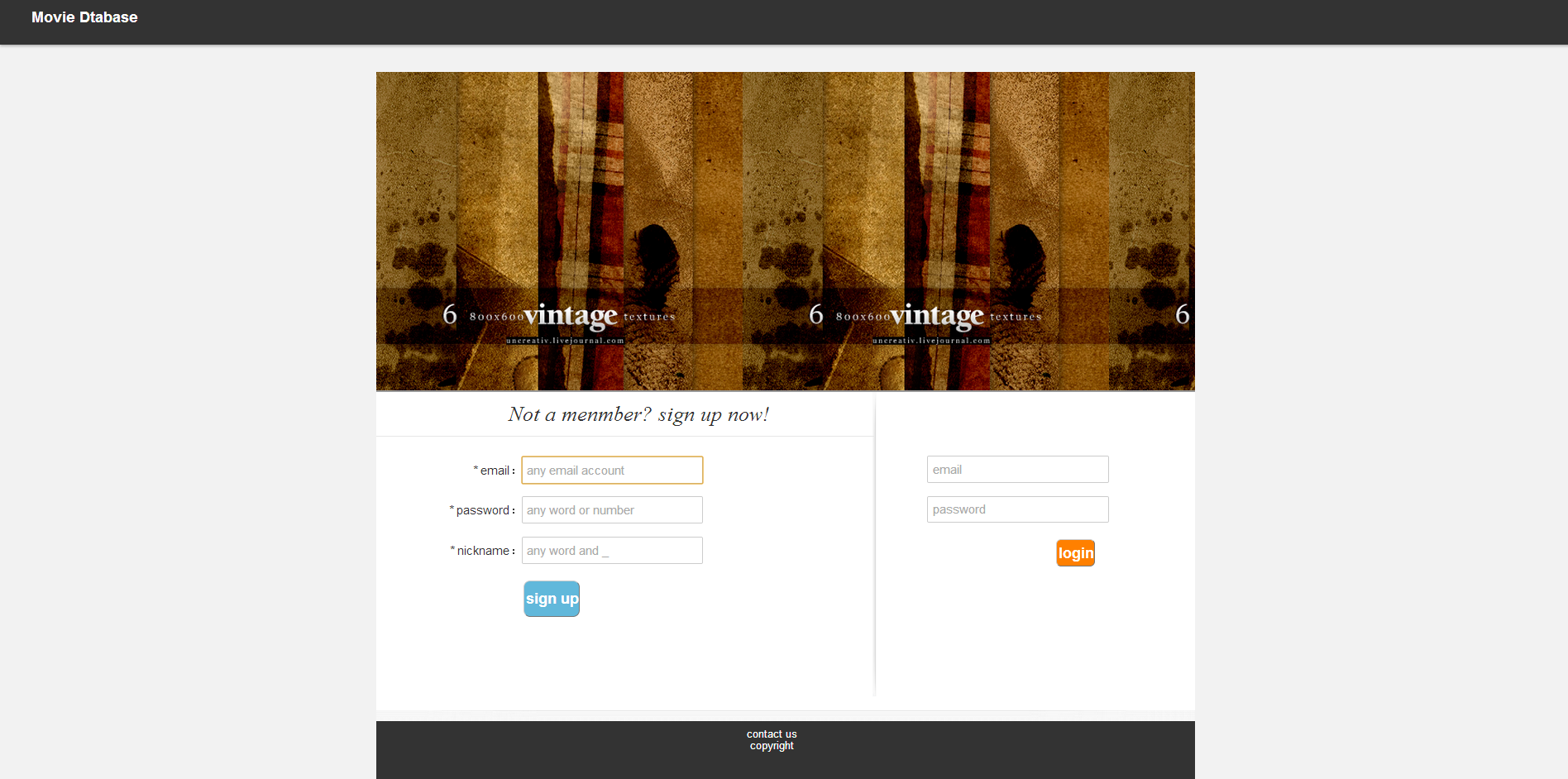
As for the other half of data, we didn’t find the appropriate API that provide the information and extract data from web page ended in failure. Considering that input data manually will be troublesome, we write a data generator to generate data randomly in the database.

# Software/Hardware platform

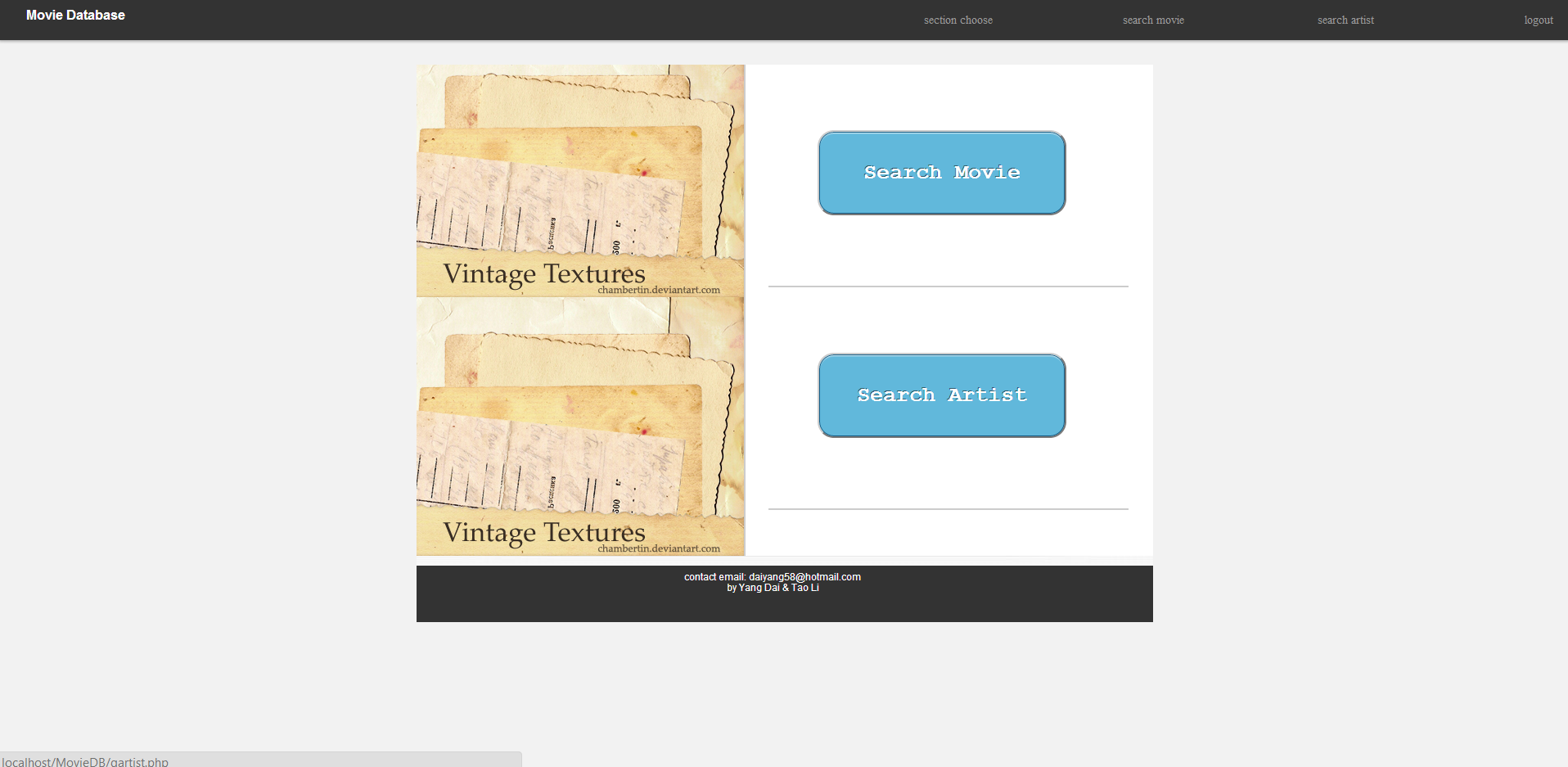
The database is running on a Lenovo desktop with window8 64bit and 8g memory. For the development convenience we utilize wampserver which is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows us to manage easily our database.

# User’s Guide

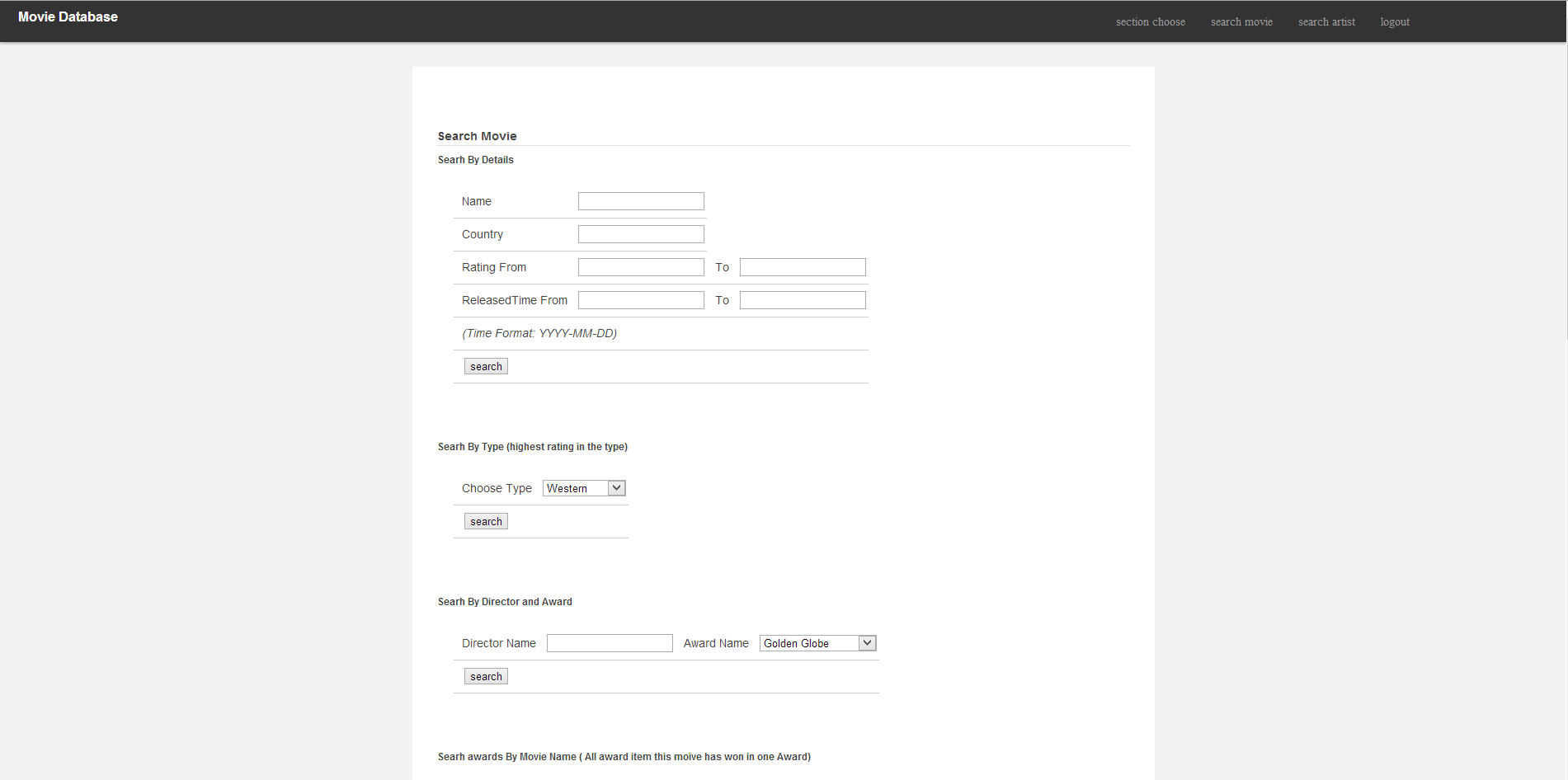
1. Starting at *intro.html* page, user first need to register or login to use the system. The intro page is as below:

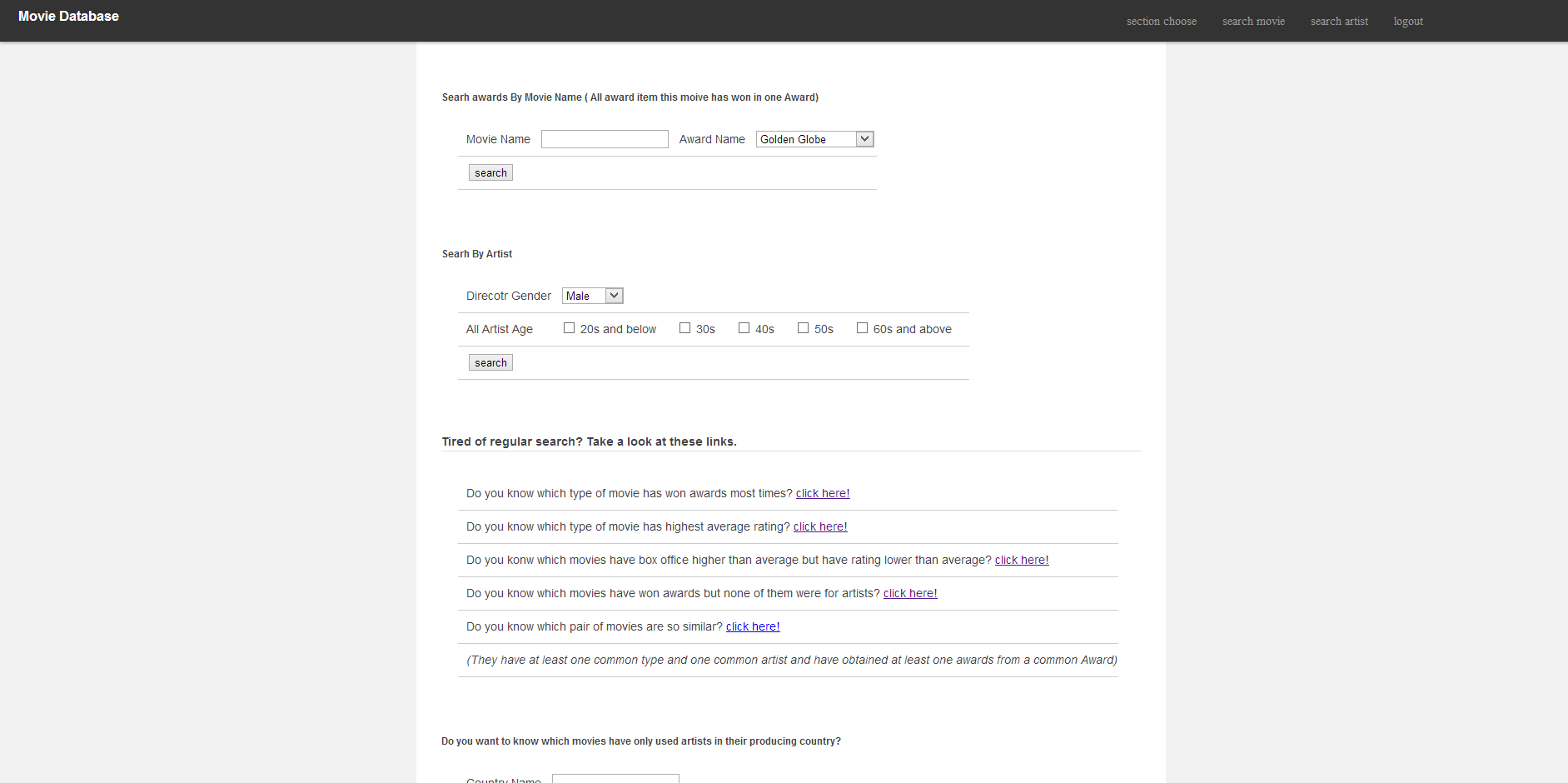


1. After login successfully, User can choose which part of function they need to use in the *module\_opt.php* page, which is displayed as below:



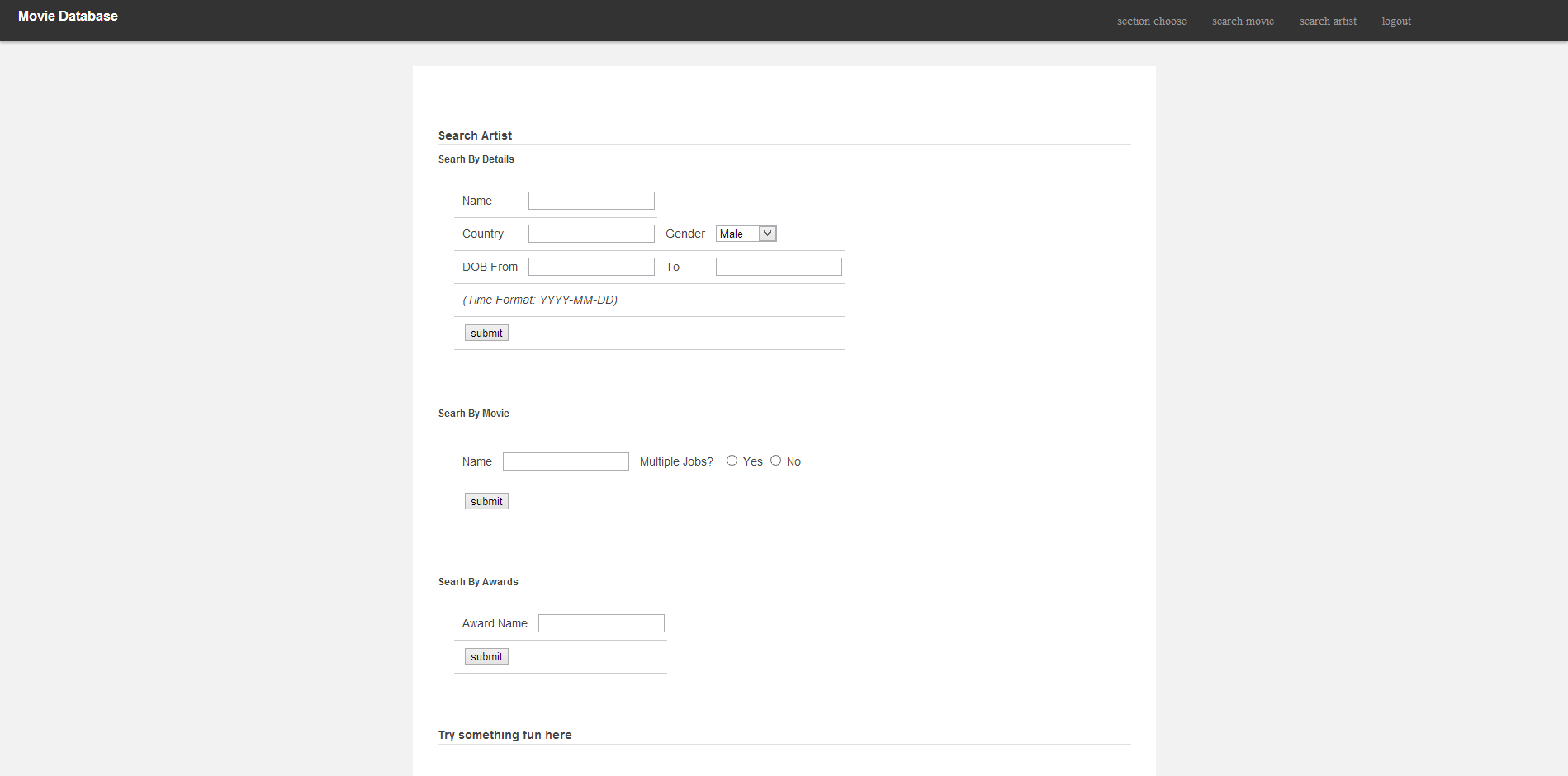
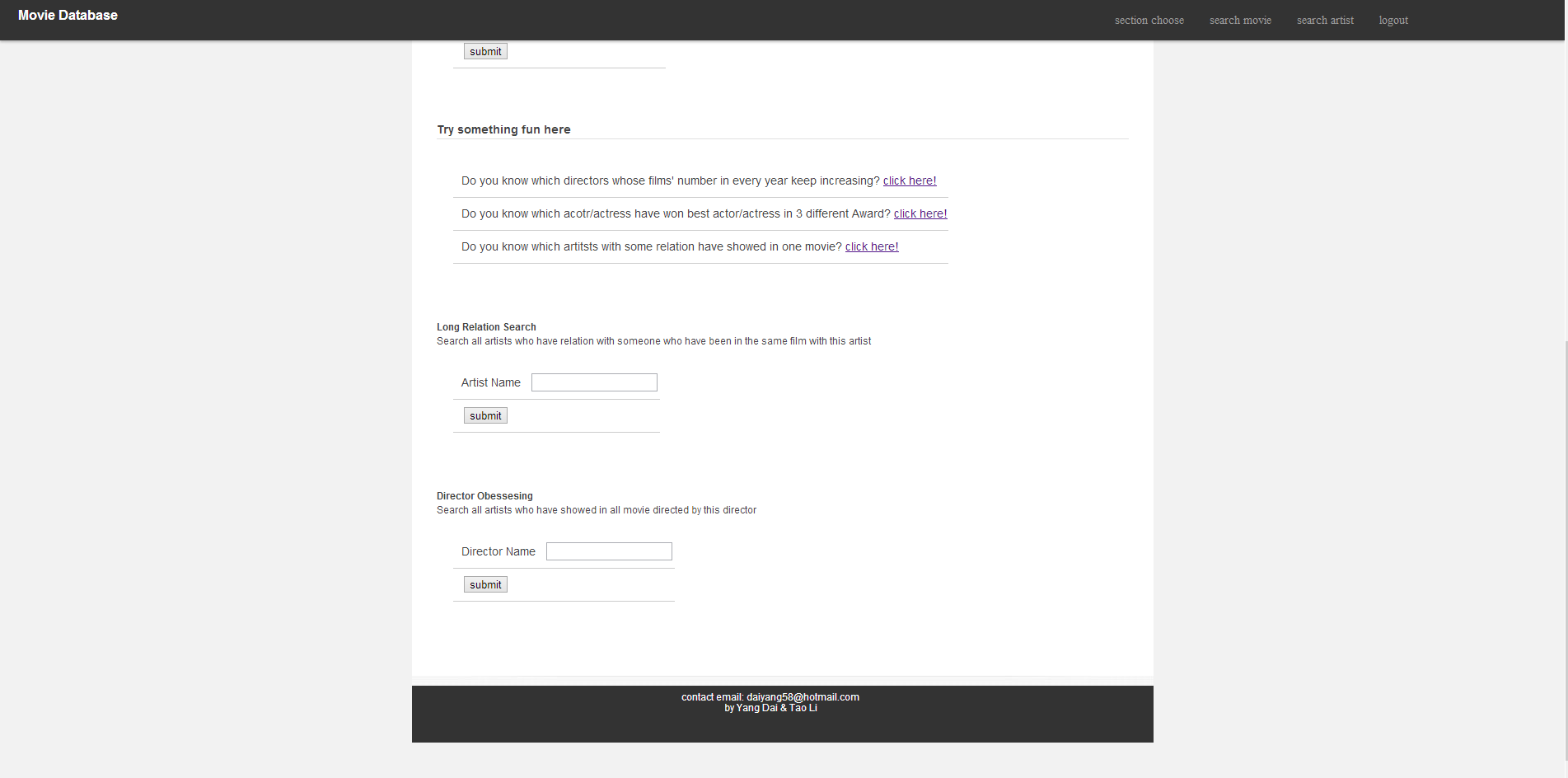
1. If user click on *Search Movie* button, they will go into *qmovie.php* page which is for movie query, displaying as below:





There are several kinds of search for movie, such as search by details, serach by artists and so on. User can choose different search according to their requirement. Also, there are some links of fun search results for user.

1. If user click on *Search Artist* button, they will go into *qartist.php* page which is for artist query, displaying as below:

Artist search page is similar to the movie search page, consisting of different search methods and some fun links. Also there are some hyper link on the top of borad for user to fast go into other pages.

# Major/Minor Areas

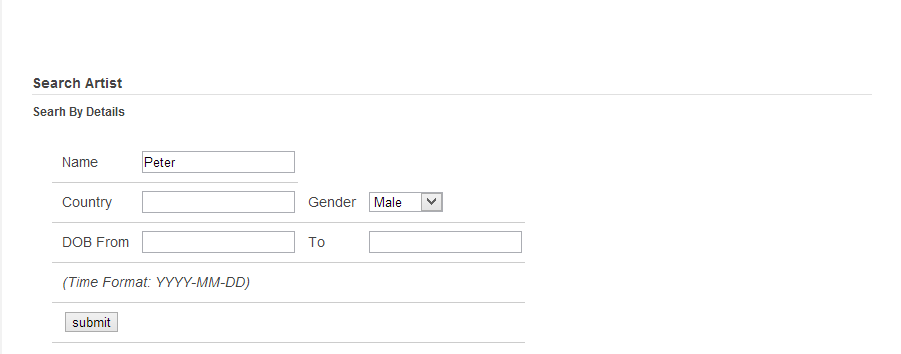
Specialization areas could include complex extraction of real data(IMDB API) and special interface. We planned to implement movie recommendation functions but failed because we were run out of time.

# Limitations

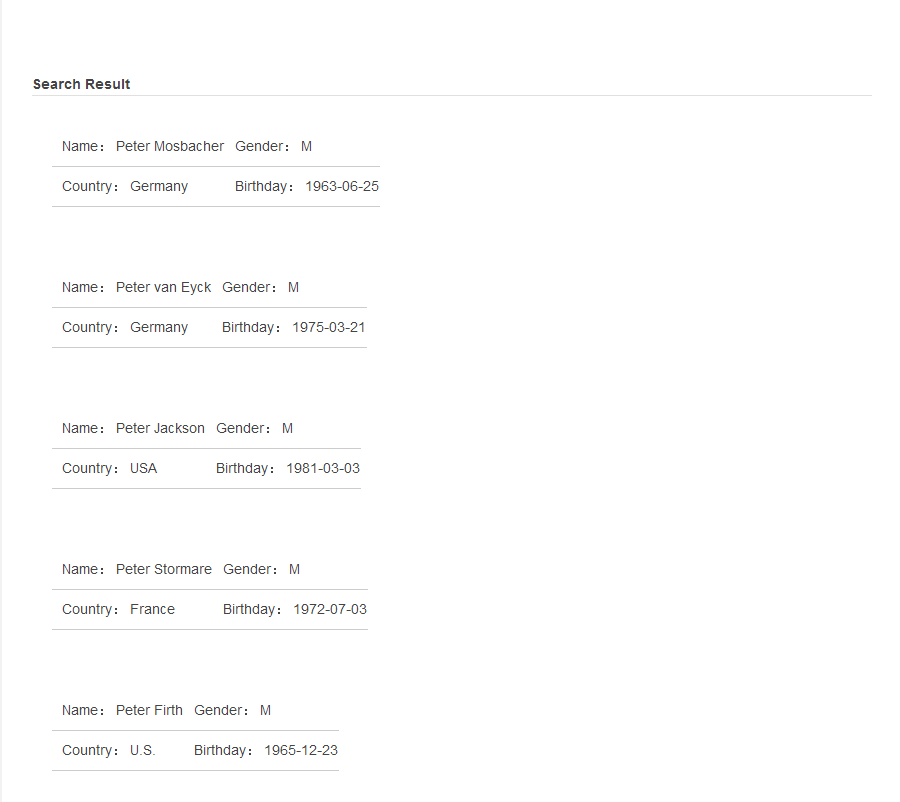
1. Not all data are real. We need find more ways to get data from internet.
2. Recommendation function is unfinished.
3. The output of search cannot be utilized again.

# Output Example

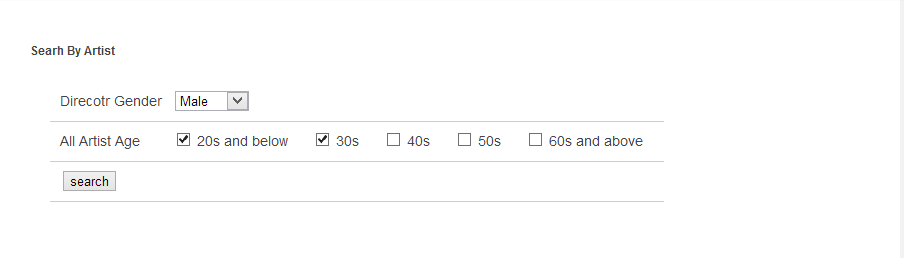
If we want to search all male artist named Peter something, the input is like following:



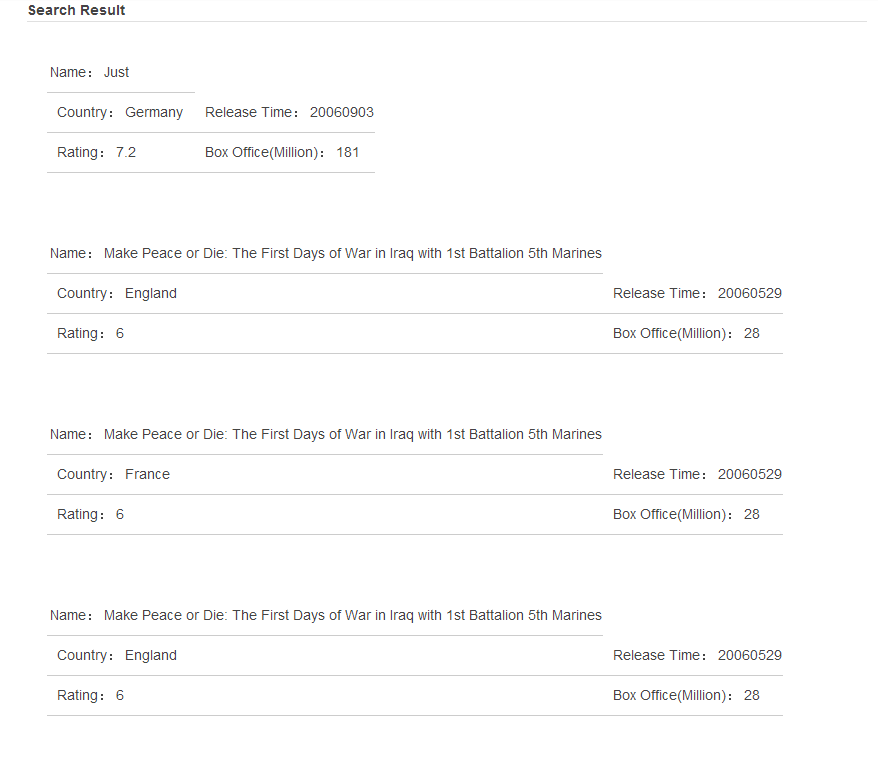
And the output will be like:



Another example. If you want to search for movies with male/female director and use artists only in some ages. Input is like below in search by artist block:



And the output will be:



# Relation Data Model

**Artist**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ArtistID | Name | Gender | Birthday | Country |
| 1641 | Lambert Hillyer | F | 1953-07-07 | Brazil |

Primary Key (ArtistID)

**Artist\_win**

|  |  |  |  |
| --- | --- | --- | --- |
| ArtistID | AwardID | MovieID | AwardTitle |
| 1642 | 3 | 273 | Best writer |

Foreign Key (ArtistID) References Artist(ArtistID)

Foreign Key (AwardID) References Award(AwardID)

Foreign Key (MovieID) References Movie(MovieID)

**Award**

|  |  |  |
| --- | --- | --- |
| AwardID | AwardName | Country |
| 7 | Oscar | USA |

Primary Key (AwardID)

**Has\_genre**

|  |  |
| --- | --- |
| MovieID | Genre |
| 273 | Western |

Foreign Key (MovieID) References Movie(MovieID)

**Has\_relation**

|  |  |  |
| --- | --- | --- |
| ArtistID1 | ArtistID2 | Relation |
| 1642 | 1955 | father |

Foreign Key (AwardID1) References Artist(ArtistID)

Foreign Key (AwardID2) References Artist(ArtistID)

**Movie**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MovieID | MovieName | Rating | Released | Country | Box\_Office |
| 277 | A Ham in a Role | 6.9 | 19491213 | USA | 241 |

Primary Key (MovieID)

**Movie\_win**

|  |  |  |  |
| --- | --- | --- | --- |
| MovieID | AwardID | Time | AwardTitle |
| 287 | 1 | 2008-09-11 | Best picture |

Foreign Key (MovieID) References Movie(MovieID)

Foreign Key (AwardID) References Award(AwardID)

**Work\_on**

|  |  |  |
| --- | --- | --- |
| MovieID | ArtistID | Job |
| 273 | 1642 | actor |

Foreign Key (MovieID) References Movie(MovieID)

Foreign Key (ArtistID) References Artist(ArtistID)

# SQL

# Find out names of all pairs of artists who appears in the same movie and have certain relation.

SELECT a1.Name AS Artist1, a2.Name Artist2

FROM Artist AS a1, Artist AS a2

WHERE a1.ArtistID < a2.ArtistID

AND EXISTS ( SELECT \*

FROM Has\_relation AS hr

WHERE hr.ArtistID1 = a1.ArtistID AND hr.ArtistID2 = a2.ArtistID

) AND EXISTS (

SELECT \* FROM Work\_on AS wo1

WHERE wo1.ArtistID =a1.ArtistID

AND wo1.MovieID in (

SELECT wo2.MovieID

FROM Work\_on AS wo2

WHERE wo2.ArtistID = a2.ArtistID

)

)

# Find out the type of movie with highest average rating.

SELECT genre

FROM Movie JOIN Has\_genre ON Movie.movieID = Has\_genre.MovieID

GROUP BY genre

HAVING avg(rating)>=all

(SELECT avg(rating)

FROM Movie NATURAL JOIN Has\_genre

GROUP BY genre);

# Find out names of artists who have relation with someone who have been in the same film with someone (input), order by name.

SELECT \*

FROM Artist AS a1

WHERE a1.ArtistID IN (

SELECT hr.ArtistID1

FROM Has\_relation AS hr

WHERE hr.ArtistID2 IN (

SELECT wo1.ArtistID

FROM Work\_on AS wo1

WHERE wo1.MovieID IN (

SELECT wo2.MovieID

FROM Work\_on AS wo2 INNER JOIN Artist AS a2 ON a2.ArtistID = wo2.ArtistID

WHERE a2.Name like "%Harry%"

)

)

) ORDER BY a1.Name DESC

# Find out movies that was directed by someone (input) and won some awards (input).

# eg. name ="Alain Lefebvre", awardname = "National film"

SELECT distinct(moviename)

FROM Artist JOIN Work\_on ON Artist.ArtistID = Work\_on.ArtistID JOIN Movie ON

Movie.movieID = Work\_on.movieID JOIN Movie\_win ON

Work\_on.movieID = Work\_on.movieID JOIN Award ON Award.awardID = Movie\_win.awardID

WHERE name ="Alain Lefebvre" and job = 'director'

and awardname = "National film"

# Find out names of artists who have been in all movie directed by someone (input).

SELECT \*

FROM Artist AS a1

WHERE a1.ArtistID NOT IN (

SELECT wo1.ArtistID

FROM Work\_on AS wo1

WHERE wo1.MovieID NOT IN (

SELECT wo2.MovieID

FROM Work\_on AS wo2

INNER JOIN Artist AS a2 ON a2.ArtistID = wo2.ArtistID

WHERE a2.Name like “%peter%” AND wo2.Job = “director”

)

)

# Find out the type of film that has won awards most times.

SELECT genre

FROM Movie JOIN Movie\_win ON Movie.MovieID = Movie\_win.MovieID JOIN Has\_genre ON

Movie.MovieID = Has\_genre.MovieID JOIN Award ON

Award.awardID = Movie\_win.awardID

GROUP BY genre

HAVING count(awardname) >= all

(SELECT count(awardname)

FROM Movie JOIN Movie\_win ON Movie.MovieID = Movie\_win.MovieID JOIN Has\_genre ON

Movie.MovieID = Has\_genre.MovieID JOIN Award ON

Award.awardID = Movie\_win.awardID

GROUP BY genre)

# Find out all the awards a film (input) won in certain Award (input).

SELECT a.AwardTitle

FROM Award AS a

INNER JOIN Movie\_win AS mw ON mw.AwardID = a.AwardID

INNER JOIN Movie as m ON m.MovieID = mw.MovieID

WHERE m.MovieName like “” AND a.AwardName like “”

# Find out the director whose films’ number in every year keep increasing.

SELECT \*

FROM

Artist as A1 JOIN Work\_on ON A1.ArtistID = Work\_on.ArtistID JOIN Movie ON

Movie.movieID = Work\_on.movieID

WHERE job = "director" and released like '2013%'

GROUP BY A1.artistID

HAVING count(Movie.movieID)>=

(SELECT count(Movie.movieID)

FROM Artist as A2 JOIN Work\_on ON A2.ArtistID = Work\_on.ArtistID JOIN Movie ON

Movie.movieID = Work\_on.movieID

WHERE job = "director" and A1.artistID = A2.artistID and released like '2012%'

GROUP BY A2.artistID

HAVING count(Movie.movieID)>=

(SELECT count(Movie.movieID)

FROM Artist as A3 JOIN Work\_on ON A3.ArtistID = Work\_on.ArtistID JOIN Movie ON

Movie.movieID = Work\_on.movieID

WHERE job = "director" and A1.artistID=A3.artistID and released like '2011%'

GROUP BY A3.artistID ))

# Find out all movies that have box office higher than average but have rating lower than average.

SELECT \*

FROM Movie AS m

WHERE m.Box\_office > (

SELECT AVG(Box\_office)

FROM Movie

)

AND m.rating < (

SELECT AVG(Rating)

FROM Movie

)

# Find out names and their jobs and the movies of all artists who have multiple job in that movie.

# eg. moviename = 'Gun Law Justice'

SELECT \*

FROM Artist JOIN Work\_on ON Artist.ArtistID = Work\_on.ArtistID JOIN Movie ON

Movie.movieID = Work\_on.movieID

WHERE Movie.moviename = 'Gun Law Justice'

GROUP BY artistID

HAVING count(job) >1

# Find out names of artists who have won “best actor/actress” in 5 different Award.

SELECT \*

FROM Artist AS a

INNER JOIN Artist\_win as awa ON awa.ArtistID = a.ArtistID WHERE awa.AwardTitle like "%best%"

GROUP BY a.Name

HAVING COUNT(awa.AwardID) >= 2

# Find out all movies have won some awards but not for actors or actress.

SELECT movieID

FROM Movie as m JOIN Movie\_win ON m.movieID = Movie\_win.movieID

WHERE m.movieID not in

(SELECT movieID

FROM Artist\_win)

# Find out all movies that have won some Awards and were directed by a female director and have used Artists no more than 20 in the movie.

SELECT m

FROM Movie AS m

INNER JOIN Movie\_win AS mw ON m.MovieID = mw.MovieID

INNER JOIN Work\_on AS wo ON wo.MovieID = mw.MovieID

INNER JOIN Artist AS a ON a.ArtistID = wo.ArtistID

WHERE a.gender = "F" AND wo.Job = "director"

AND EXISTS (

SELECT \*

FROM Artist AS a1

INNER JOIN Work\_on AS wo1 ON wo1.ArtistID = a1.ArtistID

INNER JOIN Movie as m1 ON m1.MovieID = wo1.MovieID

WHERE wo.Job != "Director"

AND FLOOR(DATEDIFF(m1.Released, a1.Birthday) / 365) between 10 and 100

)

#Find out all movies produced by some country (input) and use only artists in its country.

SELECT moviename

FROM Movie as m1

WHERE m1.movieID not in(

SELECT m2.movieID

FROM Artist as A JOIN Work\_on ON A.ArtistID = Work\_on.ArtistID JOIN Movie as m2 ON

m2.movieID = Work\_on.movieID

WHERE A.country != m1.country

)

#baisc movie search. Search by details, need to deal with null situation. PHP code is as following:

<?php

function checkMovieName($MovieName)

{

$str = "";

if($MovieName != NULL)

$str = "MovieName like '%".$MovieName."%' and ";

return $str;

}

function checkCountry($Country)

{

$str = "";

if($Country != NULL)

$str = "Country like '%".$Country."%' and ";

return $str;

}

function checkMinRating($MinRating)

{

$str = "Rating > '0'";

if($MinRating != NULL)

$str = "Rating > '".$MinRating."'";

return $str;

}

function checkMaxRating($MaxRating)

{

$str = "";

if($MaxRating != NULL)

$str = "Rating > '".$MaxRating."' and ";

return $str;

}

function checkMinRelease($MinRelease)

{

$str = "";

if($MinRelease != NULL)

$str = "Released > '".$MinRelease."' and ";

return $str;

}

function checkMaxRelease($MaxRelease)

{

$str = "";

if($MaxRelease != NULL)

$str = "Released < '".$MaxRelease."' and ";

return $str;

}

?>

<?php

if(isset($\_POST["qDetail"])) {

$MovieName = $\_POST["name"];

$Country = $\_POST["country"];

$MinRating = $\_POST["minrating"];

$MaxRating = $\_POST["maxrating"];

$MinRelease = $\_POST["minrelease"];

$MaxRelease = $\_POST["maxrelease"];

$query = mysql\_query("select \* from Movie where ".checkMovieName($MovieName).checkCountry($Country).checkMinRelease($MinRelease).checkMaxRelease($MaxRelease).

checkMaxRating($MaxRating).checkMinRating($MinRating)

)

or die("Invalid query:".mysql\_error);

}

?>