Final project report

Team member:
112550011 李佑軒, 112550153 蔡凱旭, 112550129 江秉璋, 112550164 蔣忻勳, 112550183 蔡承志

Main idea

英雄聯盟是世界上數一數二有名、熱門的電腦線上遊戲，而就像NBA、MLB或是其他競技項目一樣，數據是提升自己對遊戲理解很重要的一部分。我們希望做出一個能讓使用者透過吸收整理過後的資料，對這個遊戲更為了解，並運用於自己的遊戲經驗的工具。

其中我們的功能包含二大部分：
1. 使用者個人數據，對自己遊戲場次進行CRUD操作後，能分析資料，並對自己的遊戲狀況更加了解。
2. 資料庫數據，在從S4開始蒐集的龐大資料庫中，洞察特定英雄或對局在普遍狀況下的表現。其中我們加入，能夠篩選一些更細節的條件下數據的功能，例如特定十隻英雄對局的勝率、獲得第一條小龍後的勝率...等等。
有了這個兩個功能，使用者除了能更輕鬆掌握自己遊戲的狀況並加以調整外，還能比對其他玩家的狀況、在遊戲前、遊戲中調整策略，創造更好的勝率。

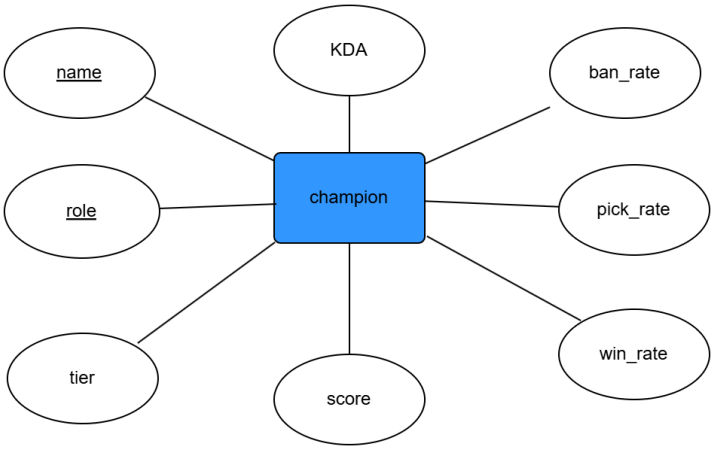
Data

1、Hero Data application
https://www.kaggle.com/datasets/vivovinco/league-of-legends-champion-stats/data
https://www.kaggle.com/datasets/vivovinco/league-of-legends-stats-s13

S12.1~S13.13的英雄詳細資料(From kaggle)
內容

- champion_stats_S12、champion_stats_S13
  - Name：英雄名稱
  - Role：英雄負責位置（職位）
  - Tier：英雄強度評級
  - Score：英雄表現分
  - Win %：英雄勝率
  - Pick %：英雄出場率
  - Ban %：英雄被禁止出場機率
  - KDA：英雄平均KDA (Kills + Assists / Death)

ER model



2、Match win rate estimate

https://www.kaggle.com/datasets/paololol/league-of-legends-ranked-matches/data?select=teamstats.csv

S4開始的180000場的積分對戰資料 (From kaggle)

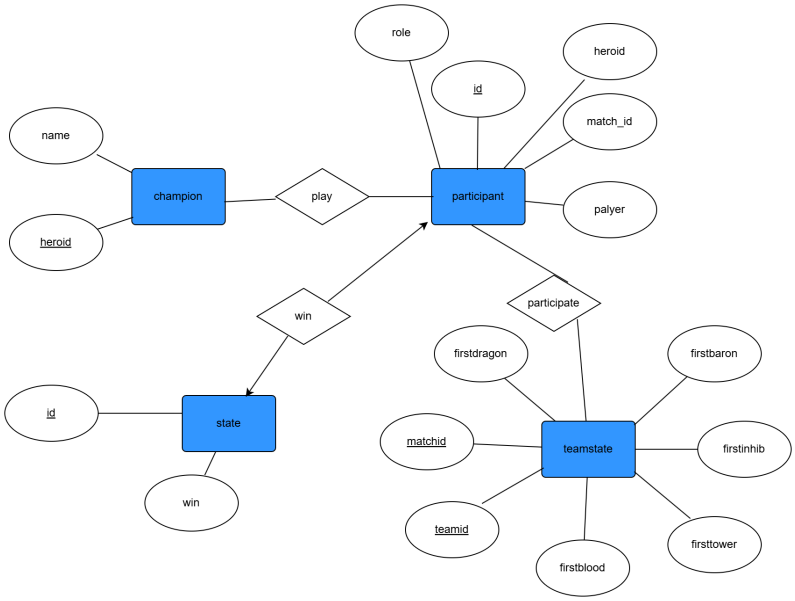
- champ:
  - name: champion names
  - id: champion id

- participants:
  - id: participant id
  - matchid: match id
  - player: player number (from 1 to 10 in each game)
  -
  - heroid: which champion the player choose
  - role: which position the player play

- state:
  - id: participant id
  - win: win or lose (0-lose 1-win)

- teamstats:
  - matchid: match id
  - teamid: team id (player 1~5 team_id = 100, player 6~10 team_id = 200)
  - firstblood: whether the team get first blood
  - firsttower: whether the team get first tower
  - firstinhib: whether the team get first inhibitor
  - firstbaron: whether the team get first baron
  - firstdragon: whether the team get first dragon

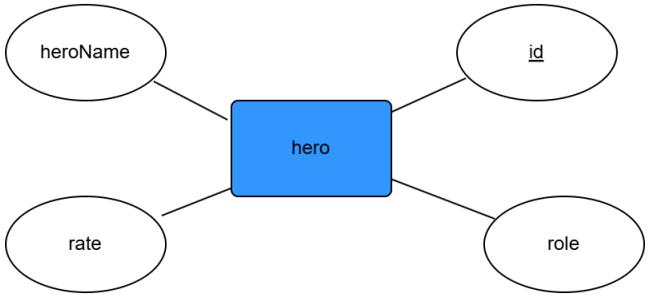
ER model



3、User's hero evaluation application (CRUD from user)

- hero_evaluation
  - ID
  - hero
  - role
  - rate

ER model



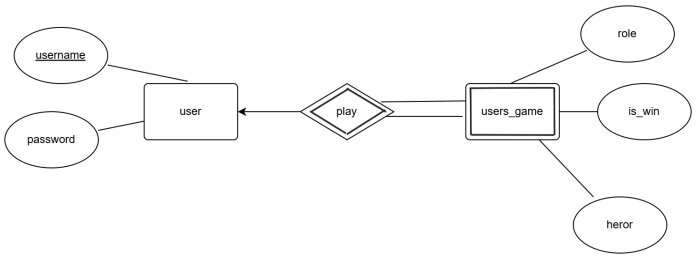
4、User signup and evaluate users' game win rate

- users
  - username
  - passwd

- users_game
  - username
  - hero
  - role

- is_win

ER model



Database

Database type : mysql

How do you maintain your database :

Hero_evalution table 部分可供使用者做CRUD的操作
/evaluation 頁面可供玩家新增一筆資料，insert進hero_evaluation table中並為此筆資料提供一個 unique 的 primary key。
query這個table可讓使用者顯示出某個路線的所有資料
可以透過unique的ID 去 update或delete這個database

users table 可以讓使用者新增自己的使用者帳號，但新增時會先比對是否惟在相同使用者，不存在才能新增這個使用者

user_game table 可以讓使用者新增一筆自己的遊戲對局資料，新增資料時要先檢查使用者給的密碼是否是正確的，才可以插入自己的資料。

12_1 ~ 13_13 tables 由kaggle上的英雄評價資料，不會對其做任何新增、刪除或更新

champ、participants、states、teamstates tables 為180000場積分對局構成，從kaggle上的資料匯入，用來提供一些勝率的查詢，也不會對此資料庫做任何新增、刪除或更新。

Describe how you connect your database to your application as detail as possible

*Using node.js for back-end server

使用mysql模組建立與資料庫連線

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'qwer@tyui@op12@',
    database: 'final_project'
});

connection.connect((err) => {
    if (err) {
        console.error('連線到 MySQL 失敗:', err.stack);
        return;
    }
    console.log('已連線到 MySQL 資料庫');
});

module.exports = connection;
```

當後端程式需要使用時引入連接的資料庫 const db = require("../db.js");
使用function：db.query(sql指令、對應的參數、callback function)來完成mysql query
當back-end 收到特定的request，會做對應的query並回傳合適的response(Application 3 有每頁詳細的API)

exception handling(也一樣詳細在Appcation 3)：
若有query錯誤或是其他問題(像是使用者已存在、密碼錯誤、找不到英雄ID等等)
會回傳對應的response({“result” : “incorrectpw”}等等)，讓前端可以處理。

防止SQL injection ：使用參數化查詢，使用者輸入的東西不會算在sql語法裡，一定是當參數處理

```
router.put("/data/:id", (req, res)=>{
    let id = Number(req.params.id);
    let payload = req.body;
    let sql = "select count(*) as cnt from hero_evaluation h where h.ID = ?";
    let respond = {};
    db.query(sql, id, (err, result)=>{
        if(err) console.log(err);
        if(result[0]["cnt"] === 0){
            respond = {"result" : "fail"};
        }
```

圖中的ID一定是當參數處理，取代sql qeury的 ‘ ? ’，防止惡意攻擊。

限制API發送：有些功能database query要跑10s左右，所以發送request後，前端會限制使用者再次發送request，直到收到response。
像是單人對戰查詢系統中，查詢後會有請稍等圖示，直到收到response。

使用try and catch：對有機會出錯的sql_query使用try and catch 指令，若query出錯，也要reponse錯誤訊息給前端，否則可能會有許多request完全沒有收到對應reponse，造成非預期影響，也沒有對應措施。

Application

1 & 2. Describe the interface of your application and the function of your application

Title page：Get access to all the fantastic application pages
下圖中每個按鍵可以跳轉到對應的功能頁面

個人查詢：
註冊使用者頁面：讓使用者可以註冊自己的帳號，供個人對局資料使用

1. 此功能分成三個輸入欄位，第一個為輸入要註冊姓名，第二個欄位輸入註冊的密碼，最後一個欄位為再次確認輸入的密碼。

2. 根據輸入的兩次密碼判斷是否相同，若相同就會將資料傳給後端，若不相同會顯示出輸入的密碼錯誤，並在兩秒後重整畫面。

3. 在兩個密碼相同的情況，會去判斷輸入的使用者姓名是否已經註冊過，註冊過會顯示使用者姓名已經註冊過，若還沒註冊過會把使用這姓名和密碼註冊，供之後使用，並在頁面上顯示註冊成功。

個人英雄資料頁面:讓使用者可以給英雄與它對應的路線一個評價

1. 新增英雄資料
   輸入英雄、選擇路線(下拉式選單)與對應的評價(1~10)
   按下新增按鈕後，可以創建一筆使用者對英雄的評價

2. 查詢英雄資料
   選擇一個路線(下拉式選單), 並按下查詢按鈕
   可以看到使用者新增的英雄評價資料
   包括資料的編號(唯一)、角色、路線、與評價

包含剛剛新增的那筆

3. 刪除英雄資料
   輸入英雄資料的唯一編號, 按下刪除鑑
   若存在那筆資料, 則將那筆資料刪除
   否則跳出找不到此ID的警告

4. 修改英雄資料
   輸入要修改資料的編號，選單選擇新的路線，再選擇新的評價
   按下修改按鍵
   若資料ID存在，修改資料成功
   否則跳出找不到此ID的警告

修改英雄資料

輸入要修改的資料編號: 4

新的路線: 中路 新的評價 (0~10): 8

修改

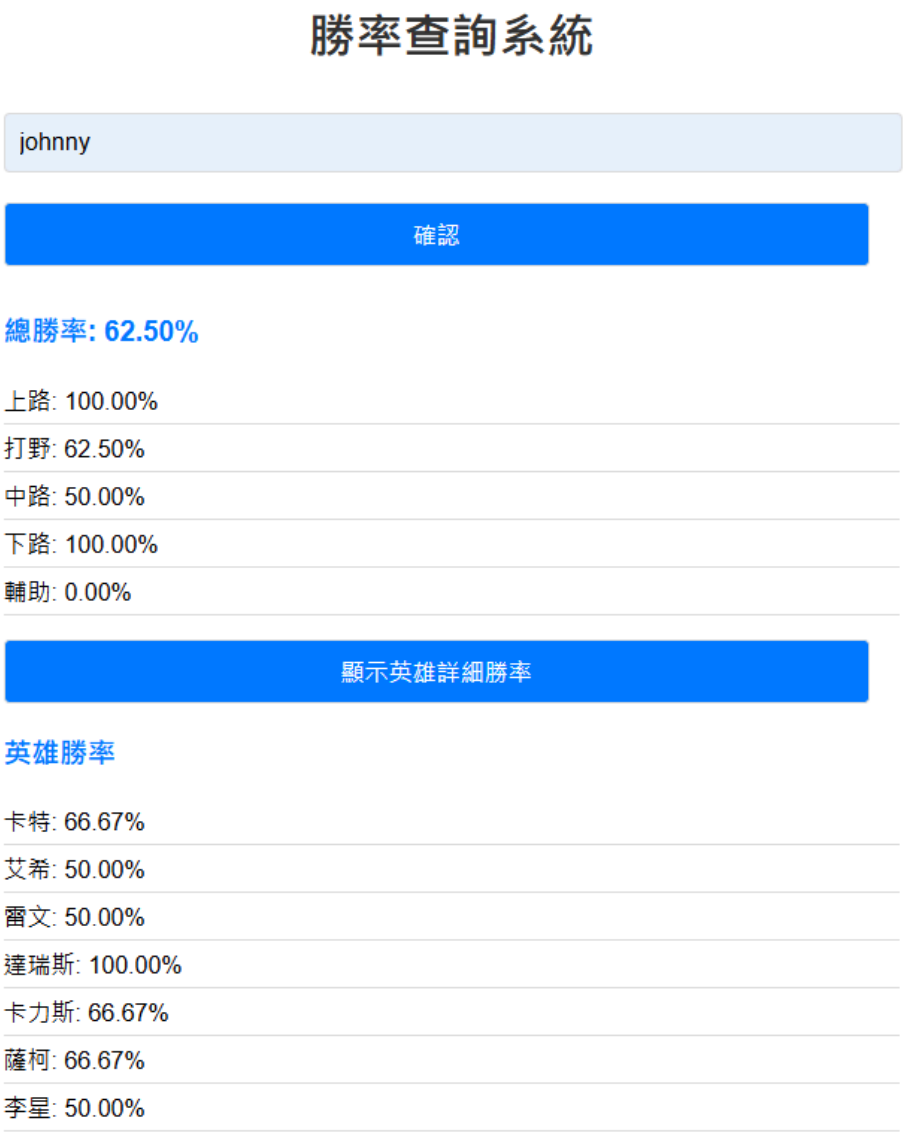
查詢英雄資料

選擇路線查詢: 中路 查詢

編號	角色	路線	評價
4	卡特蓮娜	中路	8
8	加里歐	中路	9

勝率查詢頁面：使用者可以上傳自己的對局資料，可顯示每路的勝率及角色分別的使用勝率

1. 勝率查詢系統
   輸入使用者名稱後按下確認鍵，顯示此玩家對局的總勝率以及每條路線分別的勝率
   再按下顯示英雄詳細勝率，顯示每個英雄分別的勝率



2. 新增遊戲結果資料
   輸入使用者名稱以及對英的密碼，再輸入使用英雄、路線及此戰的勝負，按下新增結果。
   若無此使用者或使用者密碼錯誤，則請使用者再試一次
   若帳號密碼皆無誤，則新增資料成功

資料庫查詢:

各賽季英雄資訊:讓使用者可以查詢特定英雄在特定賽季的各式評價

1. 查詢英雄資料:
   選選擇賽季(ex S12, S13, 下拉式選單)、版本(ex S12.1~S12.23, 下拉式選單)
   選擇英雄(ex Aatrox, 下拉式選單), 並按下搜尋按鈕
   可以看到從 Kaggle 上所下載的英雄評價資料
   包括英雄在該賽季「所有負責位置」的 Tier、Score、Win %、Pick %、Ban %、KDA 。

   若使用者未選擇賽季, 會 alert 使用者。

   Tier 和 Win% 會隨著評價好壞改變顏色。

對局資料：有以下四個功能

1. 單人對戰：使用者可以選擇兩個英雄，會顯示兩個在歷史對局有兩隻英雄對線的第一隻英雄的勝率，如果使用者選了兩個相同的英雄則會alert使用者要使用不同的英雄

2. 團隊對戰:使用者輸入己方五隻英雄和對方的五隻英雄，網站會顯示在歷史對局中這樣組合我方的勝率，若使用者選擇相同的英雄(因為是積分對戰，所以也不能和對面用一樣角色)會alert使用者。
   若在歷史對局中未找到相關資料，則會alert使用者“未找到相關資料”

團隊對戰勝率查詢

3. 英雄分析：使用者選擇一名英雄以及路線，會顯示其對線勝率最高和最低的前五隻英雄，其中對戰場數<100的會被排除在外。

4. 目標勝率：使用者選取一個特殊情況(拿到首龍、拿到首塔等)會顯示在歷史資料中符合此條件的勝率。

3. For each function of your application describe how you make it possible in detail

■ query design (including discussion on “why the query is designed like this”)

■ exception handling

*Using node.js for back-end server

1. 註冊使用者頁面**(/adduser)**

新增使用者 (post request /adduser/addNewUser)

先找到是否有使用者 : " select * from users where username = ?"

若有結果 "insert into users (username, passwd) values(?, ?)"

否則response {"result" : "already_exist"}

成功新增response {"result" : "success"} 供前端處理

2. 英雄評價頁面 **(/evaluation)**

顯示資料 (get requset /evaluation/data?role=路線)

找出路線符合的回傳給前端 : "select * from hero_evaluation h where h.role = ?"

response: {"result" : array}

element in array is object type eg: { "ID": 6, "hero": '雷文', "role": '上路', "rate": 6}

新增資料 (post request /evalation/data)

sql command : "insert into hero_evaluation (hero, role, rate) values (?, ?, ?)"

response: {"message" , "成功"} or {"message" : "插入資料失敗"}

刪除資料 (delete request /evaluation/data/:id)

先找到是否存在此ID的資料 "select count(*) as cnt from hero_evaluation h where h.ID = ?"

若不存在 response {"result" : "fail"}供前端處理

若存在刪除此ID 的資料"delete from hero_evaluation where ID = ?"

並response {"result" : "success"} 給前端

更新資料 (put request /evaluation/data/:id)

先找到是否存在此ID的資料 "select count(*) as cnt from hero_evaluation h where h.ID = ?"

若不存在 response {"result" : "fail"}供前端處理

若存在則修改此ID資料 "update hero_evaluation h set role = ?, rate = ? where h.ID = ?"

並response {"result" : "success"} 給前端

3.勝率查詢頁面 **(/winrate)**

查詢使用者勝率(get request /winrate/userWinRate?username=username)

查詢使用者每條路線的勝率以及總勝率

```
SELECT
    role,
    ROUND(AVG(is_win) * 100, 2) AS win_rate
FROM
    users_game
WHERE
    username = ?
GROUP BY
    role
```

```
SELECT
    ROUND(AVG(is_win) * 100, 2) AS total
FROM
    users_game
WHERE
    username = ?
```

回傳結果給前端

查詢使用者英雄勝率 (get request /winrate/heroWinRate?username=username)

```
SELECT
    hero,
    ROUND(AVG(is_win) * 100, 2) AS win_rate
FROM
    users_game
WHERE
    username = ?
GROUP BY
    hero
```

新增對局資料

先查詢是否有此使用者與密碼是否相符"select * from users u where u.username = ? and u.passwd = ?"

若不符合回傳{"result" : "incorrectpw"}

若符何則新增資料回傳{"result" : "success"}

4.查詢賽季英雄資料頁面 (heroData)

查詢賽季英雄資料 (get request /heroData/hero?hero=heroname&season=season)

我們為每個賽季版本都建一個獨立的table方便查詢

拿到 url 中的 hero 與 season 兩個parameter，代入下面的sql query中查詢結果

```
const querySQL = `
    SELECT
    name, role, tier, score, win_rate, pick_rate, ban_rate, kda
    FROM ${season}
    WHERE name = ?
`;
```

如果沒有資料，reponse {“result” : “NoData”}

若有資料，response {“result” : query_result} 供前端處理並呈現

query_result is array of objects，因為有些資料不只一筆，可能那隻角色有不同的position

object 格式 {name , role, tier, score, win_rate, pick_rate, ban_rate, kda}

5.對局預測勝率頁面(/match)

功能1：1v1勝率查詢 (post request /match/1v1)

Body {“hero1” : hero1_name, “hero2” : hero2_name}

後端取得兩隻英雄名字後，會代入執行一個約50行的sql_query，得到一個勝率值(平均約需執行10s)

query完的result為[{“hero_1_win_rate_percentage” : win_rate值}]，用result[0][“hero_1_win_rate_percentage”] 取得勝率的值。

後端回傳{“result” : winrate_value}，若query出問題，try 與 catch會抓到並回傳對應response

功能2：5v5勝率查詢 (post request /match/5v5)

Body = {“blue_team” : [5 heroes array], “red_team” : [5 heroes array]}

後端取得10隻英雄名字後，帶入一個65行的sql指令，得到一個勝率值(平均需執行10s)

query完的result為[{“team_1_win_rate_percentage” : win_rate值}]，用result[0][“team_1_win_rate_percentage”] 取得勝率的數字。

後端回傳{“result” : winrate_value}，若query出問題，try 與 catch會抓到並回傳對應response

功能3：英雄分析功能 (post request /match/analyze)

Body {“champion” : champion_name, “role” : champion_position }

後端取得英雄與勝率後，代入一個85行的sql指令(平均執行10s)

會得到此英雄在此路線的對手勝率最高的前五名與最低的前五名

需在資料庫中至少有100場的對局，因為有些人會選較罕見的英雄與對應路線

可能只有一兩場，影響query出來期望的結果

得到前五名與後五名後 response {“result” : array of heroes}

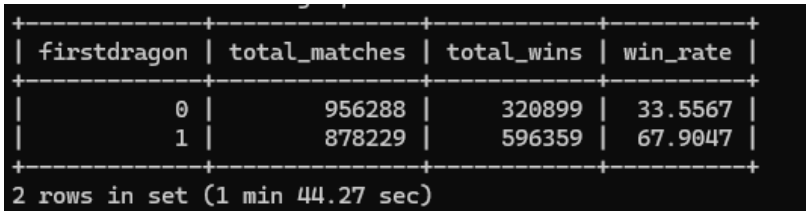
array裡的前五個是勝率最高的，後五個是勝率最低的

功能4：對局走向分析功能 (get request /match/calculate?type=?)

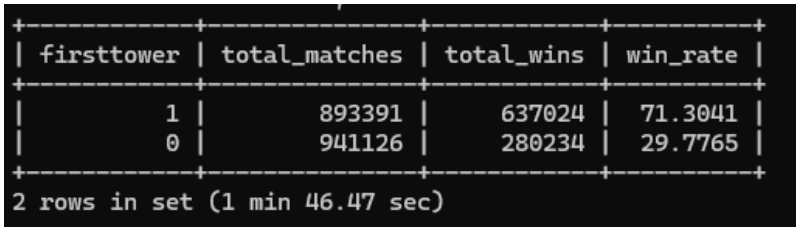
因為這裡的query都要執行大約兩分鐘，且資料庫的資料都不會變

算出來的結果都一樣，沒必要重複一直計算，所以我們決定把算出來的值直接記在後端。

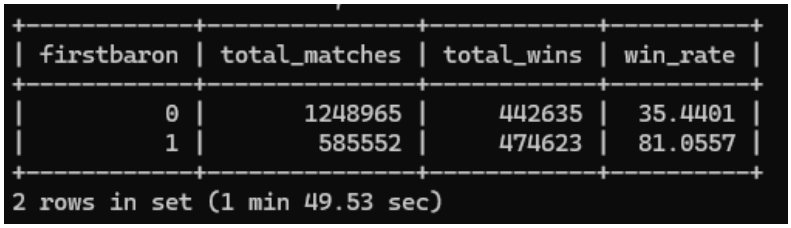
收到resquest後直接回傳對應的值。{“result” : win_rate}



```
+-------------+---------------+------------+----------+
| firstdragon | total_matches | total_wins | win_rate |
+-------------+---------------+------------+----------+
|           0 |        956288 |     320899 |  33.5567 |
|           1 |        878229 |     596359 |  67.9047 |
+-------------+---------------+------------+----------+
2 rows in set (1 min 44.27 sec)
```



```
+------------+---------------+------------+----------+
| firsttower | total_matches | total_wins | win_rate |
+------------+---------------+------------+----------+
|          1 |        893391 |     637024 |  71.3041 |
|          0 |        941126 |     280234 |  29.7765 |
+------------+---------------+------------+----------+
2 rows in set (1 min 46.47 sec)
```



```
+------------+---------------+------------+----------+
| firstbaron | total_matches | total_wins | win_rate |
+------------+---------------+------------+----------+
|          0 |       1248965 |     442635 |  35.4401 |
|          1 |        585552 |     474623 |  81.0557 |
+------------+---------------+------------+----------+
2 rows in set (1 min 49.53 sec)
```

其中三個query結果
註：因為有些對局可能沒有一方拿到baron，兩方的first_barron都會記為0
所以0的winrate會被高估，所以有加起來機率大於1的情況
但1得到的值不會有問題，所以我們就拿1的值來當回傳的value ({“result” : win_rate})

Others

Github : https://github.com/Yu-Hsuan-1220/Database-Project
Hackmd：https://hackmd.io/tHKrmo_kQq2Kq7pSCglmdQ
Video：https://youtu.be/nZD-Yheqgi8

The expected progress：
第10週： 準備資料庫
第11~12週：製作網頁前端與後端，可以互動與發送要求等等
第12~13週：對使用者(前端)發出的要求做適當回應，詢資料庫拿到正確對應的資料
第13~14週：資料如何呈現給使用者，與網頁美化

The actual progress：
第11週：資料庫準備完成
第13~15週：evaluation page 及winrate page 前端介面與後端完成
第17週：繼續完成剩下功能的前後端並更使用者友善，一些問題處理

Problem & solve

因為是第一次接觸這種前後端合作的網頁，也是第一次了解API的運作原理，花了許多時間學習包括JAVASCRIPT、HTML、CSS等程式語言，也花了時間學習怎麼用node.js建立網頁後端。

製做過程中也有遇到許多問題，像是前後端要講好API的標準，request 與 response資料的詳細格式也要交代清楚，否則會有很多問題。

有些query要執行10秒左右，使用者按下搜尋按鈕後，會沒有反應，因為response還沒回來，一般使用者遇到會覺得網站當機或沒按到按鈕，可能會再多按幾次，重複發多個request，造成後端server負擔。後來討論後決定做一個請稍後動畫，解決這個問題。

還有幾個query要跑大約兩分鐘，因為我們有1800000筆資料要一一核對，且table也很多，所以query要跑一段時間，但因為那些得出來的值都是固定的，對局資料不會有變化，沒有必要等使用者request再計算，所以我們決定直接回傳我們勝率。(像是拿到第一條小龍的勝率)

Contribution of each member:
112550011 李佑軒:evaluation與winrate功能前端網頁、所有頁面的後端server與sql
112550129 蔡凱旭:match 功能前端網頁 與 match頁面有關的sql
112550153 江秉璋:adduser 前端網頁
112550164 蔣忻勳:網站首頁連接各項功能
112550183 蔡承志:heroData 前端網頁