Data Base Report

0616059 吳炯毅
0616214 陳昱安
0616231 彭世丞
0616236 趙秉濂

I. Data

ⅰ、 在增加使用者登入與查詢功能後，我們總共有四個 table，交易紀錄(trade)、使用者資料(utilizador)、查找資料 (forselect)、查詢紀錄 (history)

a. 交易紀錄(trade)：來自 政府房價實價登錄資料 的原始資料。

欄位中文	Attribute Name	Description	Example	type
縣市	City	The city that the house located	臺北市	char(4)
鄉鎮市區	district	The villages and towns urban district	文山區	char(4)
交易標的	trade_target	transaction sign	房地(土地+建物)+車位	char(13)
土地區段位置建物區段門牌	address	land sector position building sector house number plate	臺北市萬華區康定路 1~30 號	char(40)
土地移轉總面積平方公尺	land_area	land shifting total area square meter	34.05	float
都市土地使用分區	area_use	the use zoning or compiles and checks	商	char(1)
非都市土地使用分區	nonmetro_d	the non-metropolis land use district	山坡地保育區	char(10)
非都市土地使用編定	nonmetro_u	non-metropolis land use	丙種建築用地	char(10)
交易年月日	trade_date	transaction year month and day	1081220	int
交易筆棟數	target_detail	transaction pen number (the details of “trade_target”)	土地 3 建物 1 車位 0	nvarchar(10)
移轉層次	trade_floor	shifting level	三層	nvarchar(18)
總樓層數	total_floor	total floor number	十五層	nvarchar(4)

建物型態	state	building state	套房(1 房 1 廳 1 衛)	nvarchar(20)
主要用途	purpose	main use	住家用	nvarchar(10)
主要建材	materials	main building materials	鋼筋混凝土造	nvarchar(10)
建築完成年月	building_date	year month and day of construction to complete	701209	int
建物移轉總面積平方公尺	building_area	building shifting total area	23.04	float
建物現況格局-房	room	Building present situation pattern - room	4	int
建物現況格局-廳	hall	building present situation pattern - hall	2	int
建物現況格局-衛	health	building present situation pattern - health	3	int
建物現況格局-隔間	compartment	building present situation pattern - compartmented	有	varchar(2)
有無管理組織	manage	Whether there is manages the organization or not	無	varchar(2)
總價元	price	total price NTD	6547000	int
單價元平方公尺	unit_price	the unit price (NTD / square meter)	243500	int
車位類別	berth_type	the berth category	坡道機械	varchar(4)
車位移轉總面積平方公尺	berth_area	berth shifting total area square meter	40.27	float
車位總價元	berth_price	the berth total price NTD	1200000	int
備註	note	the note of extra describing	含增建或未登記建物。;	varchar(50)
編號	trade_id	number of this transaction	RPXNMLOLKHPFFBA38CA	varchar(19)

b. 使用者資料(utilizador)：註冊時會向此表插入資料、登入時會確認是否在這個表內有相符的紀錄。

Attribute Name	Description	Example	type
userid	使用者帳號	john	VARCHAR(255)
email	使用者信箱	john@gmail.com	VARCHAR(255)
password	使用者密碼	******	VARCHAR(255)

c. 查找資料(forselect)：給予縣市資料的快取 table 從以下 query 取自 a 的 table

```
64   SELECT distinct city, district
65   from trade;
```

Attribute Name	Description	Example	type
city	縣市	臺北市	VARCHAR(4)
region	地區	士林區	VARCHAR(6)

d. 查詢紀錄(history):

Attribute Name	Description	Example	type
userid	使用者帳號	godjj	VARCHAR(255)
city	縣市	臺北市	VARCHAR(4)
district	地區	士林區	VARCHAR(6)
trade_date	交易日期	10901	int
highest_price	最高總價	1000	bigint
lowest_price	最低總價	100000000	bigint

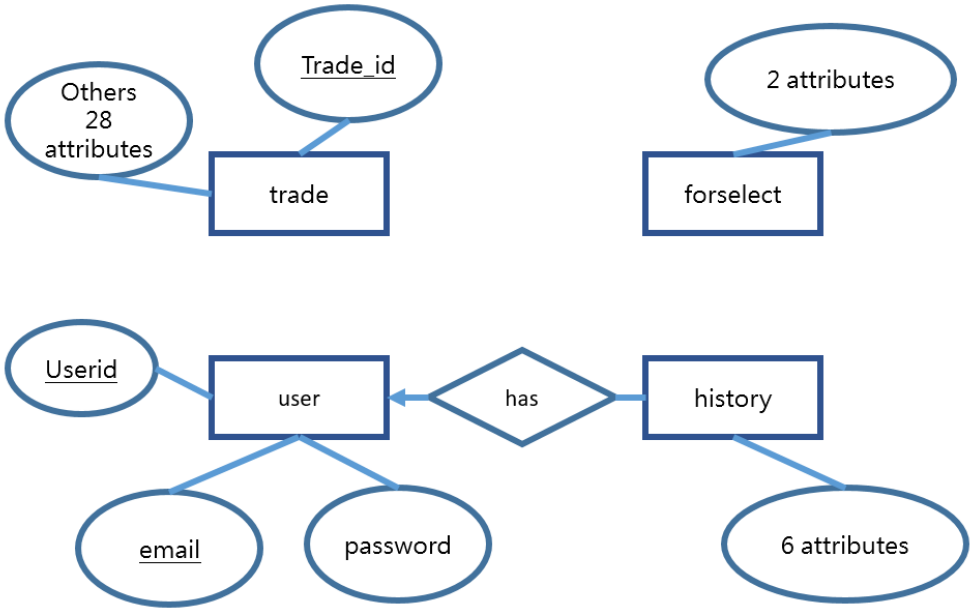
ii、 Normalize data

1. Attempt1：我們一開始直接把原始的表全部連接成一張大表，根據不同張表加入新的欄位 city (ex. A_lvr_land_A.csv 中的檔案的 city 就都會是臺北市。)

   1.1. Memory 總數：522 MB

```
+--------------------+--------------+
| table_schema       | MB           |
+--------------------+--------------+
| information_schema |   0.15625000 |
| mysql              |   2.45838070 |
| performance_schema |   0.00000000 |
| sys                |   0.01562500 |
| test_project       | 522.60937500 |
+--------------------+--------------+
5 rows in set (0.04 sec)
```

1.2. ER model



1.3. Query 速度：附錄有相對應的 query_num 與其程式碼

Test_query_num	Time used (sec)
1	1.83
2	0.73
3	0.83
4	0.72

After building index on (city, district)

Test_query_num	Time used (sec)
1	0.01
2	0.03
3	0.03
4	1.39

2. Attempt2：為了減少大表中的 NULL 的資料佔掉的空格數，我們試著把空格的欄未拆出來，做 lossless decomposition。

2.1. 一開始，我們的表就符合 2NF，因為有 trade_id 這個 primary key，且一個 row 原本就只有一筆的資訊。因此我們想要試著往 3NF 邁進，由於窮舉所有 columns 的 dependency 太多了所以我們就從 X->Y，當 X、Y 都是單一欄位的情況開始。發現除了 primary key 的 trade_id 可以推出所有的 column，沒有任兩個 column 有 dependency，連最有可能的 district->city 都發現有 7 個重名的區在不同的城市中。(附圖於後，表 1)

a、 也就是說沒有任一欄可以在相同的值的狀況下另一欄也相同 (for all a in column A, b in column B ; if a1=a2 than b1=b2 ; ai, bi are in same row) 上面這件事對我們表中的任兩欄都辦不到.

2.2. 因此，我們找出 12 個有 NULL 值存在的欄位，將他們佔有互相的空值比例算出來(ex. A, B 欄的空值比例 = (A 與 B 皆為空值的列數) / (A\B 空值列比較多的) ；如果 A 有 5 個列是 NULL，而 B 有 3 個列是 NULL，A 與 B 同時是 NULL 的有 2 列，則 A 與 B 的空值比=(2)/max(5,3)=2/5=0.4)。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		address	area_use	nonmetro_	nonmetro_	trade_floo	total_floor	purpose	materials	building_d	unit_price	berth_type	note	empty row
2	address	1	0.2	0.87	0.87	0.27	0.27	0.27	0.27	0.27	0	0.93	0.67	15
3	area_use	0	1	0.05	0.05	0.59	0.6	0.6	0.6	0.6	0.03	0.92	0.67	140625
4	nonmetro_d	0	0.01	1	1	0.18	0.18	0.2	0.18	0.2	0.02	0.62	0.69	590713
5	nonmetro_u	0	0.01	1	1	0.18	0.18	0.2	0.18	0.2	0.02	0.62	0.69	591244
6	trade_floor	0	0.44	0.57	0.57	1	1	1	1	1	0.03	0.97	0.58	188309
7	total_floor	0	0.44	0.57	0.57	1	1	1	1	1	0.03	0.97	0.58	189097
8	purpose	0	0.42	0.6	0.6	0.94	0.94	1	0.94	0.95	0.03	0.97	0.58	200467
9	materials	0	0.44	0.57	0.57	1	1	1	1	1	0.03	0.97	0.58	188403
10	building_date	0	0.43	0.59	0.59	0.94	0.94	0.95	0.94	1	0.03	0.97	0.57	200055
11	unit_price	0	0.31	0.84	0.84	0.42	0.42	0.42	0.42	0.43	1	0.27	0.63	15128
12	berth_type	0	0.26	0.74	0.75	0.37	0.37	0.4	0.37	0.4	0.01	1	0.65	489331
13	note	0	0.19	0.82	0.82	0.22	0.22	0.23	0.22	0.23	0.02	0.63	1	500500

2.3. 依照上圖的計算，可以看出有三區的空值比較聚集，因此我們把大表拆成四個小表，其欄位如下。Area_use 與 address 和 unit_pric 由於空值比較少或站的空間較少就不拆出來做處理。

a、 Trade_land_info：nonmetro_d、nonmetro_u、trade_id

b、 Trade_struct_info：trade_floor、total_floor、purpose、materials、building_date、trade_id

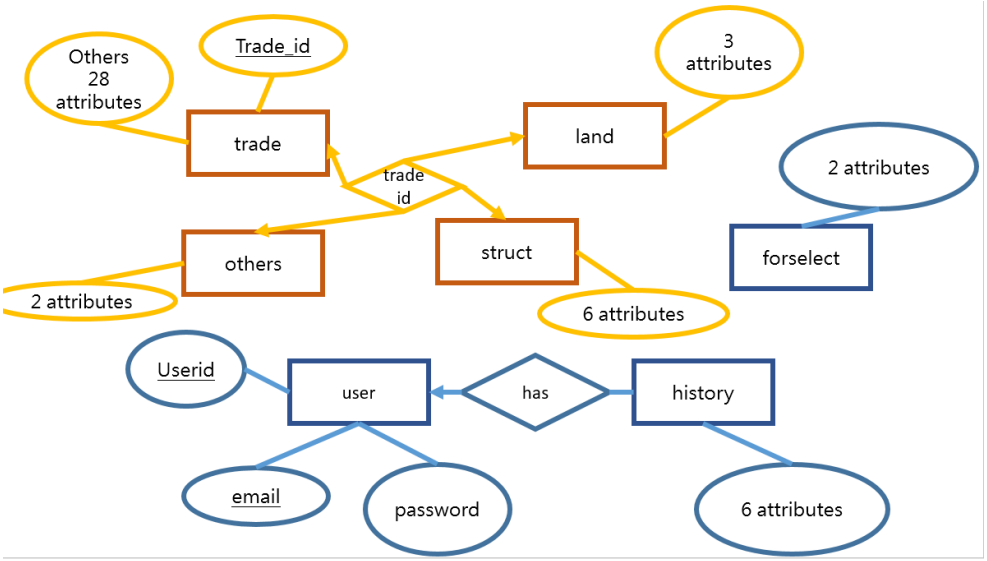
c、 Trade_others_info：berth_type、note、trade_id

d、 Trade：others attribute、trade_id

2.4. Memory 總數：644 MB，使用 index 會再增加 30MB 左右，沒有如預想的減少，可能是因為 create table 的成本 > 我們減少 NULL 的成本。

```
+--------------------+--------------+
| table_schema       | MB           |
+--------------------+--------------+
| information_schema |   0.15625000 |
| mysql              |   2.45838070 |
| performance_schema |   0.00000000 |
| sys                |   0.01562500 |
| test_project       | 644.87500000 |
+--------------------+--------------+
5 rows in set (0.04 sec)
```

2.5. ER model：黃色是拆分的部分



2.6. Query speed：

Test_query_num	Time used (sec)
1	1.9
2	0.66
3	0.71
4	0.69

After building index (city, district, yearmonth)

2、3 則使用新加的 yearmonth 來用不同的 query 達到同樣的效果

Test_query_num	Time used (sec)
1	0.01
2	0.00 (using 2.1)
3	0.00 (using 3.1)
4	1.51

3. Attemp3

3.1. 依照 column 的相依性列出表 2，表中 A 與 B 欄交界的數字代表 (select distinct A, B from trade) 表的大小，越小代表 A 與 B 越有關係。依照表的資訊，我們改成拆成下列四個表。

a、 Trade_one：trade_target、area_use、compartment、manage、num

b、 Trade_two：nonmetro_d、nonmetro_u、state、purpose、materials、hall、berth_type、num

c、 Trade_three：city、district、total_floor、room、health

d、 Trade：address、land_area、building_date、building_area、price、unit_price、berth_area、berth_price、note、trade_id、trade_date、trade_floor、target_detail、num1、num2、num3

3.2. Memory 總數 320 MB：有成功有效減少儲存的空間，減少的原因是因為用代號來減少重複儲存的資料，可以從各自 table 大小看出來。

a、 Trade_one：(88, 5)

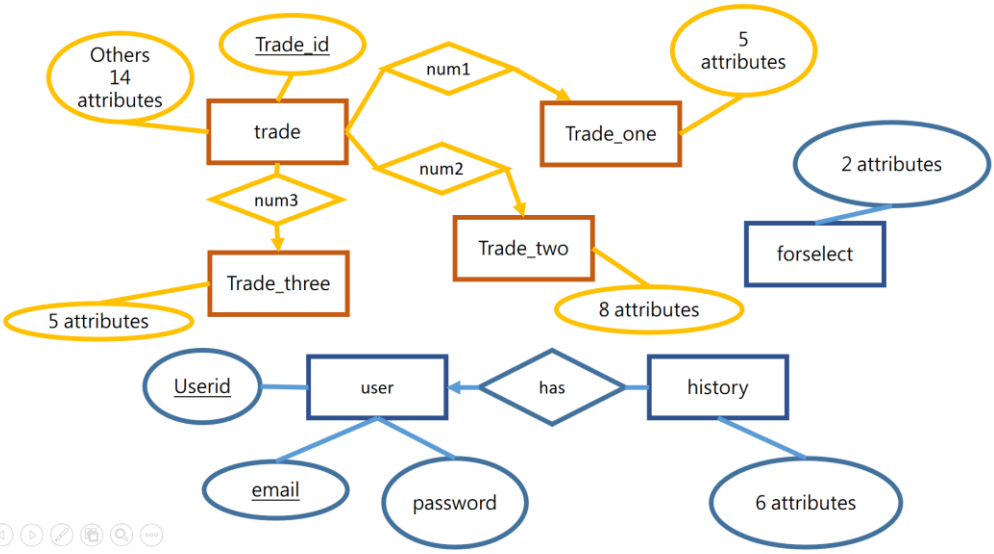
b、 Trade_two：(6194, 8)

c、 Trade_three：(43535, 5)

d、 Trade：(724391, 15)

```
mysql> select table_schema, sum((data_length+i
+--------------------+--------------+
| table_schema       | MB           |
+--------------------+--------------+
| information_schema |   0.15625000 |
| mysql              |   2.45838070 |
| performance_schema |   0.00000000 |
| sys                |   0.01562500 |
| test_project       | 320.07812500 |
+--------------------+--------------+
5 rows in set (0.14 sec)
```

3.3. ER-model：三個表利用 num 與大表連接：三個小表中的 num 都是 unique 的



3.4. 這個方法的 trade 的 num1、num2、num3 的計算需要大量的計算時間，因此我們最後還是選用 method1 做為我們 project 的使用。

II. Database

i、 DBMS

1. mysql

ii、 How to maintain

1. 有關 trade 的部分，我們需要手動將資料從網頁中載下來，經過 python 程式碼的處理再跑 sql 的 script 載入資料庫中。

1.1. data_merge.py

這裡把我們有的 198 個 csv 合在一起並做以下處理

a、 依照檔案的不同增加 city 的欄位

b、 drop 掉 trade date 小於四位的 row (1 筆)

c、 drop 掉 district 為空的 row (2 筆)

```
print(merge_df.shape)
merge_df = merge_df.loc[[len(x) >= 6 for x in merge_df['交易年月日']]]
merge_df.reset_index(drop=True, inplace=True)
print(merge_df.shape)
merge_df = merge_df[merge_df['鄉鎮市區'].notna()]
merge_df.reset_index(drop=True, inplace=True)
print(merge_df.shape)
merge_df['yearmonth'] = [ x[:-2] for x in merge_df['交易年月日']]
merge_df.rename(columns=column_dict, inplace=True)
# print(merge_df[:3])
```

```
(724394, 29)
(724393, 29)
(724391, 29)
```

1.2. Method1_init.sql

a、 在這裡 create 了 table 與 load_csv 進入 database，並建立 index 以加快搜尋的速度。

```
5
6   CREATE TABLE IF NOT EXISTS user (
7       userid VARCHAR(255) NOT NULL,
8       email VARCHAR(255) NOT NULL,
9       password VARCHAR(255) NOT NULL,
0       PRIMARY KEY (email),
1       UNIQUE INDEX email_UNIQUE (email)
2   );
3
4 > create table trade(···
5   );
6
7
8   load data local infile '../data/merge/merge.csv'
9   into table trade
0   fields terminated by ','
1   enclosed by '"'
2   lines terminated by '\n'
3   ignore 1 lines;
4
5   create INDEX city_idx on trade (city, district);
```

2. 有關 user 登入資料與其搜尋紀錄則透過網頁與 user 互動可以對資料庫做修改。

2.1. User 登入表

```
CREATE TABLE IF NOT EXISTS user (
    userid VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    PRIMARY KEY (email),
    UNIQUE INDEX email_UNIQUE (email)
);
```

code 可以參考 login.php。

login 頁面有兩個功能一個是註冊，另一個就是登入，兩個功能都是由 html 的 input 按鈕觸發。

```
<form role="form" method="post" action="./login.php">
    <fieldset>
        <div class="form-group">
            <input name="email" class="form-control" placeholder="E-mail" name="email" type="email" au
        </div>
        <div class="form-group">
            <input name="password" class="form-control" placeholder="Password" name="password" type="p
        </div>
        <input type="submit" name="login" class="btn btn-lg btn-success btn-block" value="Login" />
    </fieldset>
</form>
<form role="form" method="post" action="./login.php">
    <fieldset>
        <div class="form-group">
            <input name="userid" class="form-control" placeholder="UserId" name="userid" type="userid"
        </div>
        <div class="form-group">
            <input name="email" class="form-control" placeholder="E-mail" name="email" type="email" au
        </div>
        <div class="form-group">
            <input name="password" class="form-control" placeholder="Password" name="password" type="p
        </div>
        <input type="submit" name="register" class="btn btn-lg btn-success btn-block" value="Register"
    </fieldset>
</form>
```

```
if (isset($_POST['register']) && !empty($_POST['userid']) && !empty($_POST['email']) && !empty($_POST['password']))
```

然後一樣先檢查變數，去確認使用者是否按了按鈕。

```
$email=$mysql->real_escape_string($_POST['email']);
$password=$mysql->real_escape_string($_POST['password']);
$userid=$mysql->real_escape_string($_POST['userid']);

//test whether userid has been register
$sql = "SELECT userid FROM utilizador WHERE userid='$userid'";
$test = $mysql->query($sql);
```

然後因為我們不允許有相同 userid 的存在，所以我們會先對是否有同樣的 userid 做一次 query。

```php
if($test->num_rows == 0){
    $sql = "insert into utilizador(userid,email,password) values ('$userid','$email','$password')";
    $mysql->query($sql);
    $_SESSION['userid']=$userid;

    header("location: ./index.php");
    exit;
}
else{
    $repeat = 1;
}
```

檢查 query 傳回來的資料是不是 0 筆，如果是的話，就將新的 userid, email, password 寫入資料庫，如果不是 0 筆的話，就會將 repeat 這個變數設成 1，然後下面的程式會檢查$repeat 變數，提醒是用者這個 userid 被使用過了。

```php
if (isset($_POST['login']) && !empty($_POST['email']) && !empty($_POST['password'])) {

    $email=$mysql->real_escape_string($_POST['email']);
    $password=$mysql->real_escape_string($_POST['password']);

    $login=$mysql->query("SELECT * FROM utilizador WHERE email='$email' AND password='$password'");

    if($login->num_rows==1){
        //session_register("userid");
        $row = $login->fetch_assoc();
        $_SESSION['userid']=$row['userid'];
        //$_SESSION['userid']=$_POST['email'];
        header("location: ./index.php");
        exit;
    }
    else{
        $error=1;
    }
}
```

login 的部分也是用檢查變數的方式，去確認使用者是否點選按鈕，因為登入沒有重複的問題所以，不用像剛剛需要做兩次 query，這次只要做一次 query 並確認，傳回來的資料是否為 1 筆，是的話就登入成功，不是的話就會將$error 設成 1，下面的 php 會檢查$error 然後提醒使用者。

2.2. History 維護的方式

code 可以參考 application 中 implement detail 的 history.php 介紹。

iii、 How to connect your database to your application

1. How to connect to database

在 config.php 中:

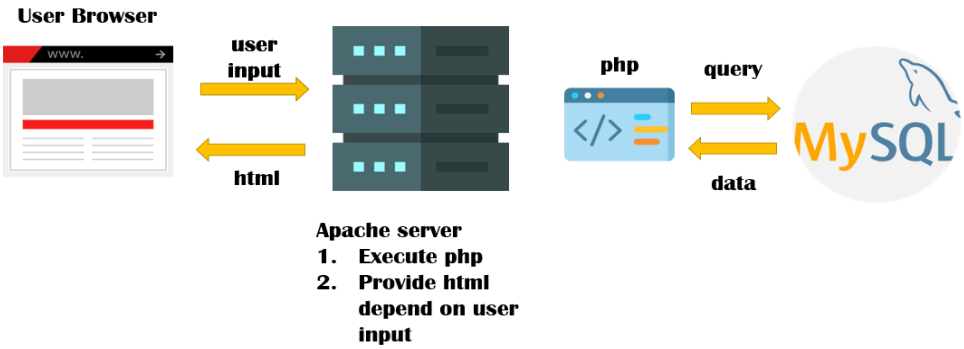
```php
<?php

session_start();

$db_host = "localhost";
$db_user = "root";
$db_pass = "";
$db_name = "test_project";

$mysql = new mysqli($db_host, $db_user, $db_pass, $db_name);
$mysql->set_charset("utf8");

?>
```

可以用 php 對 mysql database 中的 test_project 這個 database 開一個 connection，而 config.php 也可以用 include 的方式用於其他 php file 中(ex. search.php)。

III. Application

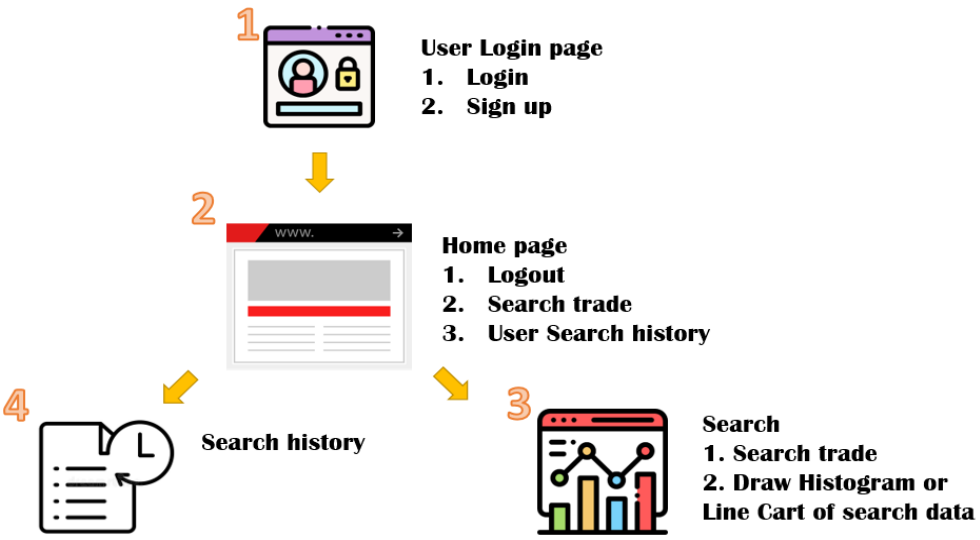
i、 Interface

我們是透過 apache server 幫我們執行 php 動態的改變網頁來與使用者互動，其角色如下



ii、 Function

我們先大致看一下我們的網頁結構，再依序依照圖上的編號一一做介紹。



1. User Login page

1.1. 登入頁面實際的圖如下

1.2. 功能：登入、註冊

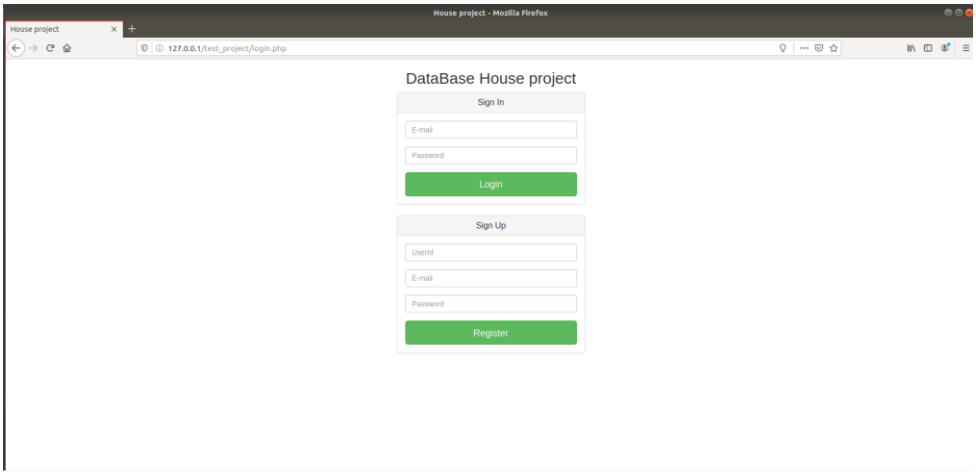
註冊後即會直接登入，下次用同一帳號登入，就會保留上次的搜尋紀錄

註冊有防止下列狀況

a、 不合格式的 email

b、 已經有同樣的 user_id 存在

c、 已經有同樣的 email 存在



2. Home page

2.1. 首頁可以連結到兩個功能，或登出

2.2. 右上方會顯示登入的使用者 id

3. Search page

3.1. 功能

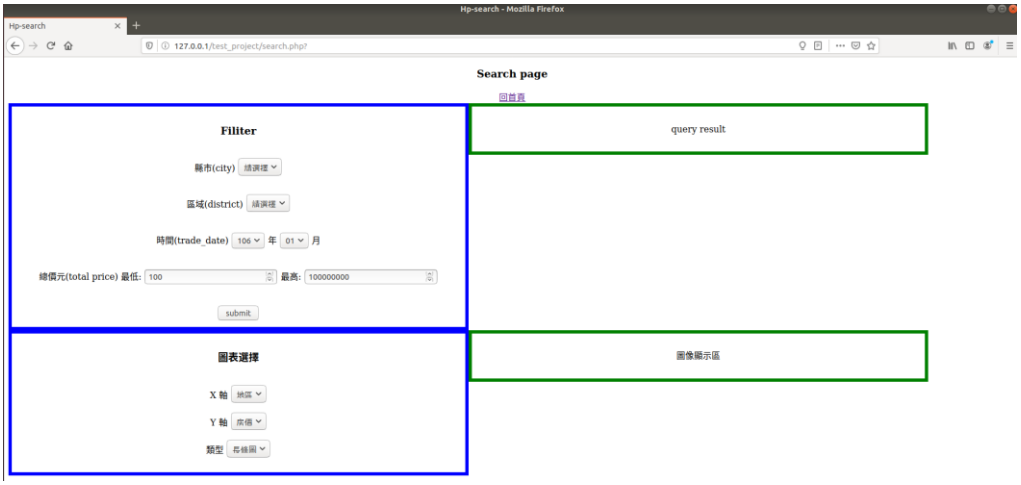
a、 區域選項會跟著縣市做轉換

b、 不填入代表不限制這個條件

c、 填好條件後右方會出現相對應的 query 與其資料筆數與資料詳情 (前 10 筆)

i. 同時，搜尋的條件會存入搜尋紀錄中

d、 限制價格的條件要在 100 ~ 10^10 之間



搜尋新竹縣峨眉鄉 108 年 1 月的交易資料

4. Search History

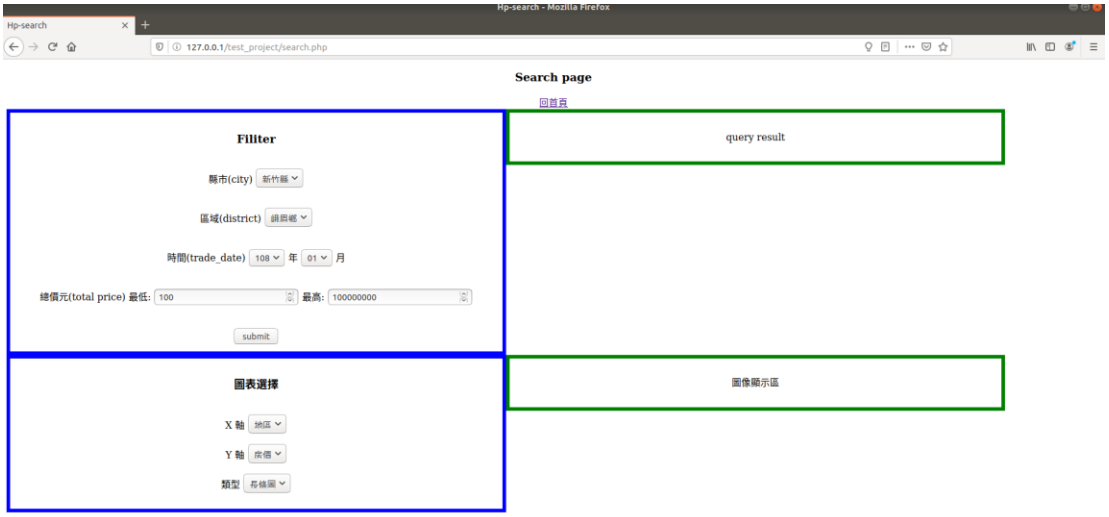
4.1. 功能

a、 搜尋紀錄的列表

b、 點前往後可以回到 - Search Page 並自動填入條件

c、 可以刪除搜尋紀錄，一個 user 限制最多 20 條紀錄

上面有剛剛搜尋的新竹縣峨眉鄉的紀錄，點按前往後會到下圖中，並自動填入條件以供搜尋或調整。



iii、 Implement detail

1. 如何做尋找資料的功能

在 search.php 中:

```php
<h3>Filiter</h3>
<form action="" method="post">
<p>縣市(city)
    <select id="myParentSelect" name="city">
    <option value="">請選擇</option>
        <?php
        // 取得第一層option資料
        include("config.php");
        $query = "SELECT distinct city FROM forselect;";
        $result = $mysql->query($query);

        //echo '<option value="taipei">taipei</option>"\n"';
        while ($row = $result->fetch_assoc() ) {
            echo '<option value="' . $row["city"] . '">' . $row["city"] . '</option>' . "\n";
            }
        ?>
    </select>
</p>
```

我們 include config.php 開啟一個 connection，然後用 SELECT 將台灣的各個縣市名稱拿到這裡，並且放進 [city] 這個下拉選單中的選項之中。

```php
<p>區域(district)
    <select id="myFirstChildSelect" name="district">
        <option value="">請選擇</option>
    </select>
</p>
```

這裡先設定一個 [district] 的下拉選單，而選項的內容會在下面的 script 填入，不同的縣市有不同的區域名稱，實作方式是用 jQuery 去判斷 city 這個欄位是否改變，一旦 city 欄位發生改變，就會先清空 district。

```javascript
$('#myParentSelect').change(function() {
    //更動第一層時第二層清空
    $('#myFirstChildSelect').empty().append("<option value=''>請選擇</option>");
```

然後將 city 資料當作參數傳給 action.php ， action.php 會由接收到的資料去從 database 中去搜尋，這個 city 有哪些 district ，然後用 json encode 再傳回來。然後再由 action.php 傳回來的資料，將他們放到 district 的選項當中。

```
$.ajax({
    type: "GET",
    url: "action.php",
    data: {
        lv: $('#myParentSelect option:selected').val()
    },
    datatype: "json",
    success: function(result) {
        //當第一層回到預設值時，第二層回到預設位置
        console.log(result);
        console.log(result.length);
        console.log(typeof result);
    if (result == "") {
        $('#myFirstChildSelect').val($('option:first').val());
    }
    //依據第一層回傳的值去改變第二層的內容
    result = JSON.parse(result);
    console.log(result);
    console.log(typeof result[0]['region']);
    while (i < result.length) {
        $("#myFirstChildSelect").append("<option value='" + result[i]['region'] + "'>" + result[i]['region'] + "</option>");
        i++;
    }
    //alert('Successfully called');

    },
```

action.php

```
include("config.php");
//$lvnum = $mysql->real_escape_string("Taipei");
$lvnum = $mysql->real_escape_string($_GET['lv']);
//$lvnum = $_GET['lv'];
//echo $lvnum;
//$lvnum = "5.5";
//echo $lvnum;
$jarray = array(); //使用array儲存結果，再以json_encode一次回傳

$query = "SELECT region FROM forselect where city='$lvnum'";
$result = $mysql->query($query);
//echo "<br>";
//echo $result->num_rows;
//echo "<br>";
if($result->num_rows > 0){
   while ($row = $result->fetch_assoc() ) {
  //關於mysql_fetch_*請參考php.net
       $jarray[] = $row;
      // echo $row['region'];
     //echo "match";
}
}
else{
    echo 0;
    return;
}
//echo $jarray;
//print_r($jarray);
echo json_encode($jarray);
return;
?>
```

```
<p>時間(trade_date)
    <select name="year">
        <option value="106">106</option>
        <option value="107">107</option>
        <option value="108">108</option>
        <option value="109">109</option>
    </select>
    年
    <select name="month">
        <option value="01">01</option>
        <option value="02">02</option>
        <option value="03">03</option>
        <option value="04">04</option>
        <option value="05">05</option>
        <option value="06">06</option>
        <option value="07">07</option>
        <option value="08">08</option>
        <option value="09">09</option>
        <option value="10">10</option>
        <option value="11">11</option>
        <option value="12">12</option>
    </select>
    月
</p>
```

這裡是 [year] 和 [month] 的下拉選單，可以讓使用者選擇查詢哪一年哪一個月分的交易紀錄。

```
<p>總價元(total price)
    最低:
    <input type="number" id="price_lb" name="price_lb" value="100" min="100" max="10000000000">
    最高:
    <input type="number" id="price_ub" name="price_ub" value="100000000" min="100" max="10000000000">
</p>
```

這個是 [price_lb] 與 [price_ub] 的選填欄位，分別代表查詢時交易紀錄的的最低與最高總價，其中最低預設為 100，最高預設為 100000000，欄位的最低限制為 100，最高限制為 10000000000。

```
<input type="submit" value="submit">
</form>
```

當使用者選擇限制條件後，有一個 [submit] 按鈕可以按，而這個按鈕對應的是 form action 的 post。也就是說，當使用者按下 [submit]，就可以透過 $_POST["xxx"] 的方法去拿使用者選擇的欄位內容。

Filiter

縣市(city) 臺北市

區域(district) 中正區

時間(trade_date) 106 年 12 月

總價元(total price) 最低: 10000 最高: 100000000

submit

而這個表單在網頁會如上圖所示。

```php
<?php
include("config.php");
$where_clause = "WHERE address LIKE \"".$_POST["city"].$_POST["district"]."%\" AND trade_date LIKE \"".$_POST["year"].
echo "where clause: ".$where_clause."<br>";

$query_cnt = "SELECT COUNT(*) as cnt FROM trade ".$where_clause.";";
$cnt = $mysql->query($query_cnt)->fetch_assoc();
echo "共 ".$cnt["cnt"]." 筆資料<br>";

$query = "SELECT address, trade_date, price FROM trade ".$where_clause." LIMIT 10;";
$result = $mysql->query($query);
if ($result->num_rows > 0) {
    echo "<table><tr><th>Address</th><th>Trade date</th><th>Price</th></tr>";
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["address"]."</td><td>".$row["trade_date"]."</td><td>".$row["price"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
?>
```

運用 php 的字串處理，將 post 的欄位內容放在相對應的 where clause 之中。這個 where clause 的限制條件包括「縣市」、「地區」、「交易時間」、「總價」(這裡為了讓 SELECT 的條件更清楚，也將 where clause 呈現在網頁上)。接著用建立起來的 connection 向 mysql database 的 trade 這個 table 作 query，分別得出符合限制的資料筆數以及符合限制的十筆資料。而 SELECT 的欄位包括「地址」、「交易時間」、「總價」。最後在將這些資料用 php 處理成表格呈現在網頁上。

query result

where clause: WHERE address LIKE "臺北市中正區%" AND trade_date LIKE "10612%" AND price BETWEEN 10000 AND 100000000

共 88 筆資料

Address	Trade date	Price
臺北市中正區中華路二段311巷31~60號	1061206	3310000
臺北市中正區中華路二段313巷31~60號	1061206	3310000
臺北市中正區汀州路二段181~210號	1061201	1400000
臺北市中正區南昌路一段1~30號	1061205	1900000

使用者所獲得的資料如上圖所示(對應先前表單的限制條件)。

2. 如何做到紀錄使用者的搜尋紀錄

search.php

```php
if (isset($_POST['search'])) {

    $city=$mysql->real_escape_string($_POST['city']);
    $district=$mysql->real_escape_string($_POST['district']);
    $trade_date=$mysql->real_escape_string($_POST["year"].$_POST["month"]);
    $trade_date=(int) $trade_date;
    $highest_price=$mysql->real_escape_string($_POST["price_ub"]);
    $highest_price=(int) $highest_price;
    $lowest_price=$mysql->real_escape_string($_POST["price_lb"]);
    $lowest_price=(int) $lowest_price;
    $userid=$_SESSION['userid'];

    //test whether userid has been register
    $sql = "insert into history (userid, city, district, trade_date, highest_price, lowest_price) values (
    $mysql->query($sql);
    unset($_SESSION['city']);
    unset($_SESSION['district']);
    unset($_SESSION['trade_date']);
    unset($_SESSION['highest_price']);
    unset($_SESSION['lowest_price']);
}
```

```html
<form action="" method="post">
    <input type="submit" name="search" value="submit"/>
</form>
```

主要是利用<form>標籤加上 submit 按鈕，讓每次是用者按下按鈕時透過 POST 的方式傳送給 server，然後 server 就會以 search 這個變數是否為空去判斷使用者是否按下按鈕，如果按下按鈕，那他就會讀取其他 city, district, year, month, price_ub, price_lb 的 variable，並經過一些格式的處理，接著就用 insert 的方式把使用者的搜尋紀錄，記錄到名為 history 的 table。

history.php

history 的頁面如上圖，他記錄使用者所有搜尋紀錄，以方便使用者再次搜尋相同資料，紀錄是在使用者搜尋的同時寫入 table 的，在這裡的兩個按鈕，分別是前往跟刪除，前往就是直接跳到 search 的頁面，並同時幫使用者輸入好資料，而刪除就是，從資料庫刪除這個紀錄。

```php
for($i = 0; $i <= 20; $i+=1){
  $temp = (string) $i;
  if( isset( $_POST[$temp] ) ){
    $userid = $_SESSION['userid'];
    $sql = "SELECT city, district, trade_date, highest_price, lowest_price FROM history WHERE userid = '$userid'";
    $result = $mysql->query($sql);

    for($j = 0; $j <= $result->num_rows; $j += 1){
      //fetch_assoc()
 $row = $result->fetch_assoc();
 if($j == $i){
      $_SESSION['city'] = $row['city'];
      $_SESSION['district'] = $row['district'];
      $_SESSION['trade_date'] = $row['trade_date'];
      $_SESSION['highest_price'] = $row['highest_price'];
      $_SESSION['lowest_price'] = $row['lowest_price'];
      break;
    }
  }

    header("location: ./search.php");
    exit;
  }
```

前往的部分是利用 for 迴圈，來確認使用者到底是按了哪一筆紀錄，確認了使用者是按了哪一筆紀錄後，就將資料記錄到$_SESSION 然後轉跳到 search 的頁面，search 的頁面會檢查$_SESSION 的變數是否存在，存在的話他就會選取，如果不存在，功能就跟之前一樣。

按上圖的第一筆紀錄的前往，就會直接跳到 search 並選好如上圖。

返回

縣市(city)	區域(district)	時間(trade_date)	最高價格(highest_price)	最低價格(lowest_price)	連結	刪除
台北市	內湖區	10701	100000000	100	前往	刪除
基隆市	仁愛區	10801	100000000	132	前往	刪除
桃園縣	復興區	10601	100000000	10055	前往	刪除

刪除的按鈕是會刪除這筆紀錄。

```
if( isset( $_POST[$temp] ) ){


  $userid = $_SESSION['userid'];
   $sql = "SELECT city, district, trade_date, highest_price, lowest_price FROM history WHERE userid = '$userid'";
   $result = $mysql->query($sql);

   for($j = 0; $j <= $result->num_rows; $j += 1){
     //fetch_assoc()
      $row = $result->fetch_assoc();
if($j == $i){
$city = $row['city'];
      $district = $row['district'];

$trade_date = $row['trade_date'];
$trade_date = (int) $trade_date;

$highest_price = $row['highest_price'];
$highest_price = (int) $highest_price;

$lowest_price = $row['lowest_price'];
$lowest_price = (int) $lowest_price;

    $sql = "DELETE from  history WHERE userid = '$userid' AND city='$city' AND district='$district' AND trade_date=$trade_date AND highest_price=$
    $result = $mysql->query($sql);
    break;
}
```

實作的方式也是透過，for 迴圈去確定使用者點選了哪一筆資料的刪除，確認是哪一筆會就用 delete 去刪除 history 這個 table 中的這筆資料。

3. How to draw the graph

graph 的部份分成長條圖和折線圖，因為其實兩者的概念及 code 都類似，這裡就以如何畫出長條圖的部分做說明。

另外，真正網頁上的是要與使用者的輸入做結合來判斷要畫出怎麼樣的圖表給使用者，但這裡先不討論使用者輸入的部分，就先單純以畫出圖表的部分做說明。

我們畫 graph 使用了 jquery 和 chartjs，所以首先在 html 中 link 它們

```
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" src="js/Chart.min.js"></script>
```

以 bar(長條圖)為例，先在 html 中給予 chart 一塊 canvas，接著 link 自己要畫出 bar 的 js 檔

```
<div id="chart-container">
    <canvas id="barcanvas"></canvas>
</div>
<script type="text/javascript" src="js/bargraph.js"></script>
```

另外，新建一個 php 檔(例如這裡取名為 bar_data.php)，並在裡頭處理 query 以及對 query 之後的資料做處理，並用 print json_encode 的方式丟出去。

```
$query = sprintf("SELECT district, COUNT(*) as cnt FROM test GROUP BY district");

$result = $mysqli->query($query);

$data = array();
foreach ($result as $row) {
    $data[] = $row;
}

$result->close();

$mysqli->close();

print json_encode($data);
```

從 php 中丟出的 data 就如同以下的形式：

[{"district":"\u4e2d\u5c71\u5340","cnt":"55"},{"district":"\u4e2d\u6b63\u5340","cnt":"21"},{"district":"\u4fe1\u7fa9\u5340","cnt":"22"},{"district":"\u5167\u6e56\u5340","cnt":"64"},{"district":"\u5317\u6295\u5340","cnt":"53"},{"district":"\u5357\u6e2f\u5340","cnt":"14"},{"district":"\u58eb\u6797\u5340","cnt":"38"},{"district":"\u5927\u540c\u5340","cnt":"25"},{"district":"\u5927\u5b89\u5340","cnt":"40"},{"district":"\u6587\u5c71\u5340","cnt":"84"},{"district":"\u677e\u5c71\u5340","cnt":"26"},{"district":"\u842c\u83ef\u5340","cnt":"23"}]

以這裡為例的話是丟出針對台北市 各個區域的交易數作統計 data

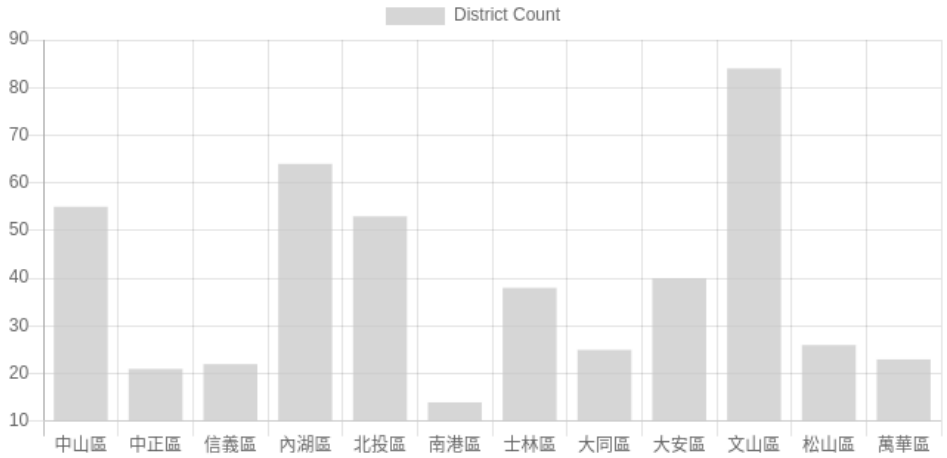
在 bargraph.js 中，首先從我們取名為 bar_data 的 php 中把 query 後處理過的 data 用 GET 的方式取得，然後把 x 軸和 y 軸的資料分別存在 js 這邊的兩個 array

```
$(document).ready(function(){
    $.ajax({
        url: "http://localhost/graph/bar_data.php",
        method: "GET",
        success: function(data) {
            var datajson = JSON.parse(data);
            console.log(data);
            var district = [];
            var count = [];
            for(var i in datajson) {
                district.push(datajson[i].district);
                count.push(datajson[i].cnt);
            }
```

最後利用 chartjs 的 Chart 來建圖並把從 php 中取得的資料餵給它，並在 datasets 中設定一些圖表內的參數，這樣就完成一張圖表了！

```
var chartdata = {
    labels: district,
    datasets : [
        {
            label: 'District Count',
            backgroundColor: 'rgba(200, 200, 200, 0.75)',
            borderColor: 'rgba(200, 200, 200, 0.75)',
            hoverBackgroundColor: 'rgba(200, 200, 200, 1)',
            hoverBorderColor: 'rgba(200, 200, 200, 1)',
            data: count
        }
    ]
};

var ctx = $("#barcanvas");

var barGraph = new Chart(ctx, {
    type: 'bar',
    data: chartdata
});
```

以長條圖並針對台北市各個區域的交易數作統計為例的話，結果如下圖



那麼若以長條圖並針對台北市各個區域的平均單價元平方公尺為例的話，結果如下圖

IV. Others

i、 Progress

1. 之前在安排進度的時候有許多任務沒有考慮到，這是我們實際進度的甘特圖

任務＼時間	5/1 -5/8	5/8 -5/15	5/15 -5/22	5/22 -5/29	5/29 -6/6	6/6 -7/6	7/6 -7/15
決定 load 的格式，成功 load csv							
學習 PHP							
設計前端樣式							
按照 function 寫出相對應 query							
撰寫前端 PHP (PHP 與 sql 做連結)							
嘗試對資料庫做正規化							
功能測試、 增加進階功能							

ii、 Encounter problems

1. 台 vs. 臺

因為台灣的很多縣市名都有「臺」這個字，例如臺北。不過因為我們在房價時價登錄網取得的資料沒有完全統一「台」與「臺」，例如資料中有「台中市」也有「臺中市」。所以為了解決這個問題，我們在建立 forselect table 時統一只保留「臺」的縣市名稱，而不考慮 trade table 中有「台」的少數情況。

這部分可以參考 data_merge.py:

```
city_dict = {
    'A':'臺北市', 'J':'新竹縣', 'S':'高雄縣',
    'B':'臺中市', 'K':'苗栗縣', 'T':'屏東縣',
    'C':'基隆市', 'L':'臺中縣', 'U':'花蓮縣',
    'D':'臺南市', 'M':'南投縣', 'V':'臺東縣',
    'E':'高雄市', 'N':'彰化縣', 'W':'金門縣',
    'F':'臺北縣', 'O':'新竹市', 'X':'澎湖縣',
    'G':'宜蘭縣', 'P':'雲林縣', 'Y':'陽明山',
    'H':'桃園縣', 'Q':'嘉義縣', 'Z':'連江縣',
    'I':'嘉義市', 'R':'臺南縣'}
```

這樣一來，在 search.php 的搜尋選項中就能夠統一用「臺」作搜尋。

2. The path to 3NF

2.1. 由於 column 太多，沒有辦法完全達到 3NF

最後使用 attempt 3 盡量以達 3NF 的精神，來做表的拆分。

iii、 Contribution of each team member

1. 趙秉濂

1.1. 寫 python 做 raw data 的前處理

1.2. 寫 database 有關的 code (ex. method1_init.sql) 與報告，並與討論可行方向。

1.3. 寫一開始 search page 靜態網頁的模板

2. 彭世丞

2.1 用 php 處理 query 後丟資料給建圖表的 js

2.2 利用 chartjs 畫出圖表

3. 陳昱安

3.1. 寫首頁靜態網頁的模板

3.2. 寫搜尋頁面(search.php)的搜尋表單以及 where clause 的處理

3.3. 寫搜尋紀錄頁面(history)靜態網頁的模板

3.4. 修正其他 php file 的 bug (ex. 跳轉頁面的路徑修正)

4. 吳炯毅

4.1 登入和登出功能(login.php logout.php)

4.2 搜尋頁面的下拉選單的 query 操作和不同選擇跑出不同選項的功能

4.3 搜尋頁面的紀錄功能

4.4 紀錄頁面(history.php)的前往功能跟刪除紀錄的功能

iv、 Repo link

1. This is our github repo link

v、 Discussion link

1. This is our hackmd discussion link

V. 附錄

i、 Test_Query

1. 產生 forselect 的 query

```
-- 1
SELECT distinct city, district
from trade
;
```

2. 算符合筆數的 query

```
-- 2
select count(*) as cnt
from trade
where   city = "臺中市" AND
        district = "東區" AND
        trade_date LIKE "10601%" AND
        price BETWEEN 100 AND 100000000
;
```

3. 列出想要資料的 query

```
17
18  -- 3
19
20  select address, price
21  from trade
22  where   city = "臺中市" AND
23          district = "東區" AND
24          trade_date LIKE "10601%" AND
25          price BETWEEN 100 AND 100000000
26  limit 10
27  ;
```

4. 算出畫直方圖所需要的 query

```
29  -- 4
30  select district, avg(price)
31  from (
32          select district, price
33          from trade
34          where   city = "臺中市" AND
35                  trade_date LIKE "10601%" AND
36                  price BETWEEN 100 AND 100000000) as sub
37  group by district
38  ;
```

ii、 附圖

1. 表一

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1		trade_id	address	unit_pric	price	land_area	building_	note	building_	berth_are	trade_dat	berth_pri	trade_flo	target_de	district	room	health	total_floo	hall	purpose	materials	city	nonmetro	state	nonmetro	berth_typ	trade_tar	area_use	compartm	manage
2	trade_id		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
3	address			FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	unit_price				FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	price					FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	land_area						FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	building_area							FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	note								FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	building_date									FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	berth_area										FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
11	trade_date											FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
12	berth_price												FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	trade_floor													FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	target_detail														FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	district															FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	room																FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
17	health																	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	total_floor																		FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	hall																			FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
20	purpose																				FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
21	materials																					FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
22	city																						FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
23	nonmetro_u																							FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
24	state																								FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
25	nonmetro_d																									FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
26	berth_type																										FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
27	trade_target																											FALSE	FALSE	FALSE	FALSE	FALSE
28	area_use																												FALSE	FALSE	FALSE	FALSE
29	compartment																													FALSE	FALSE	FALSE
30	manage																															

2.

表二

	address	land_area	area_use	nonmetro_d	nonmetro	trade_dat	target_detail	trade_flo	total_floor	state	purpose	materials	building_	building_	room	hall	health	compartment	manage	price	unit_pric	berth_typ	berth_are	berth_pri	note	trade_id	min
city	214449	219334	130	164	263	28678	2930	3049	547	249	248	271	104239	207510	620	268	591	43	43	142148	466070	153	21274	3309	42125	724391	43
district	216155	376613	1789	1557	2520	184453	12306	11214	4123	2533	2357	2444	138889	319551	3968	2054	3547	668	617	270038	614823	1330	29997	9455	54047	724391	617
trade_target	226369	110241	27	42	50	8590	746	1364	176	43	76	72	45165	81858	145	71	139	10	10	103788	272946	30	11821	1574	39146	724391	10
address(X)		493904	223265	213117	215676	632644	283859	352187	234692	233463	227103	218629	268400	432878	274926	250077	259212	212553	222640	647210	666836	229583	238241	241972	305415	724391	212553
land_area			132599	123696	121180	647279	188524	204294	188988	127105	133388	133113	391614	435587	141380	121365	139103	86826	92131	614786	671583	100392	148153	132832	196383	724391	86826
area_use				22	34	10425	1609	1824	221	71	105	113	59417	107900	257	116	244	12	12	114554	296731	47	15122	2141	40714	724391	12
nonmetro_d					107	10385	1371	1389	179	98	119	136	45654	77854	281	119	267	23	24	107822	235902	54	9455	1331	39800	724391	23
nonmetro_u						11835	1487	1385	192	112	132	164	45656	77532	297	135	282	34	34	107802	232467	57	9463	1317	39870	724391	34
trade_date							45062	64164	43470	14901	14125	14756	443964	513999	17534	10355	16159	4795	5100	471212	681087	11458	141015	60712	82045	724391	4795
target_detail								6263	3092	1779	1709	1559	90252	149936	2126	1496	2022	930	940	143263	398642	1114	18845	4281	45882	724391	930
trade_floor									3473	2001	2086	1842	163039	277686	3036	2097	2698	1192	1228	166033	467343	1725	60436	9813	49316	724391	1192
total_floor										340	443	346	94580	219563	709	356	638	101	103	153364	468280	287	28649	6059	44077	724391	101
state											173	184	77506	128407	386	179	357	24	24	114180	351445	81	17857	2345	40832	724391	24
purpose												218	64840	115815	420	208	384	47	48	111633	307212	117	17469	2567	40338	724391	47
materials													52461	101056	381	191	339	47	42	112293	280637	95	14078	2247	39966	724391	42
building_date														359142	101075	70799	90557	38390	44768	509805	550757	52263	66984	62593	119208	724391	38390
building_area															137895	103960	124581	63992	81164	569497	542709	97144	123734	120196	161317	724391	63992
room																374	471	80	124	114279	369584	201	22509	2534	42112	724391	80
hall																	360	42	59	104997	291818	99	16293	1896	40084	724391	42
health																		86	117	112579	348094	178	19030	2297	41377	724391	86
compartment																			4	95351	184020	16	10501	1371	37712	724391	4
manage																				97604	242042	16	11163	1394	38425	724391	16
price																					652678	103990	227193	139190	168082	724391	103990
unit_price																						261522	298251	258780	324026	724391	258780
berth_type																							13290	1817	39078	724391	1817
berth_area																								38993	56828	724391	38993
berth_price																									42087	724391	42087
note																										724391	0