

我的MYSQL

此書簡易介紹MYSQL操作教學



[日期]

公司地址：奧蘭尼公國

作者：

Yellowe，奧蘭尼公國國民，現職：灰之影傭兵團團員

此書介紹

關聯式資料庫MYSQL操作手冊

Mysql 操作

第一章、登入

首先，我們需啟動終端機，透過以下 SQL 指令執行登入動作：

**mysql -h host -u user –p**



-h 後面輸入您的your\_host 代表 MySQL 伺服器的主機名稱，而 –u輸入您的your\_user 則是 MySQL 帳戶的使用者名稱。如果您在執行 MySQL 的同一台電腦上登錄，則可以省略主機，只需使用下列命令：

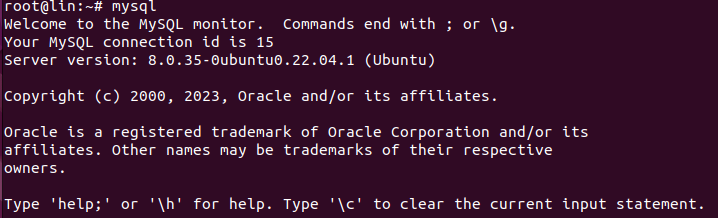
**mysql -u user –p**



登入時收到錯誤訊息，例如 ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)，則表示 MySQL 伺服器守護程序(Unix) 或服務(Windows) 未運作。請諮詢管理員或參閱官方：(https://dev.mysql.com/doc/refman/8.0/en/common-errors.html)提供的文本安裝和升級 MySQL 。

如需嘗試登入時經常遇到的其他問題的協助，請參閱官方：https://dev.mysql.com/doc/refman/8.0/en/common-errors.html。程式時的常見錯誤

您也可以直接輸入 mysql。若成功登入，結果如下，即可進行 MySQL 指令操作：



成功連線後，您可以透過QUIT、\q或exit;，三種方式來中斷連線mysql>：在 Lunix 上，您也可以按 Control+D 來中斷連線。







本章節結束前複習一下

* 登入方式

**指令：mysql -h host -u user –p，**需要your\_host 和 your\_user**，**執行 MySQL 的同一台電腦上登錄，則可以省略主機，只需使用下列命令：**，mysql -u user –p**或是直接輸入MYSQL

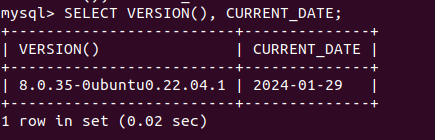
* 登出方式

QUIT、\q或exit;，三種方式

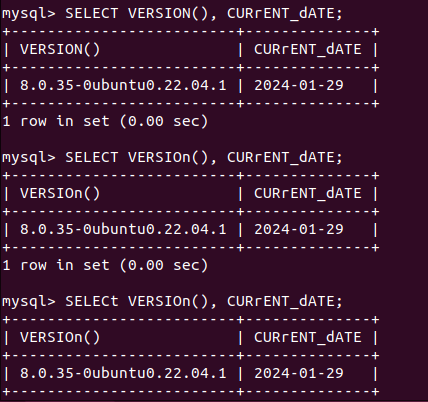
第二章、查詢

當我們希望了解 MySQL 的版本和當天日期時，我們可以透過以下 SQL 指令來進行查詢：

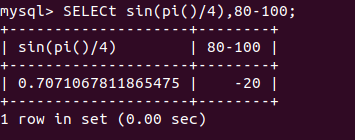
**SELECT VERSION(), CURRENT\_DATE;**

這條指令的執行結果將呈現類似下面的資訊：

指令不分大小寫皆可得到同樣結果



MYSQL也可以當作一個簡單計算機使用

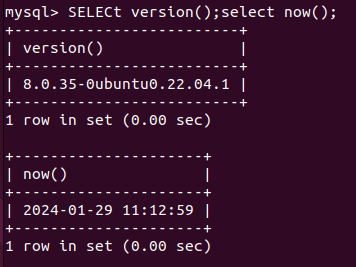


當我們希望結果會出現上下非並排，我們可以透過以下 SQL 指令來進行查詢：

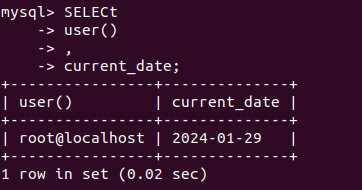
輸入

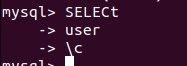
**SELECT VERSION(); SELECT NOW();**

結果會出現上下並排的結果



查詢訊息如果嫌太長 ，可以方段查詢，需要注意的是結果不能有分號。如果不想執行，直接輸入\C就能跳出回本模式。





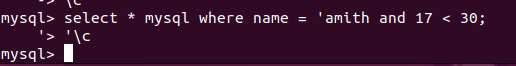
mysql>

>符號前面代表之意思

| **符號** | **意思** |
| --- | --- |
| mysql> | 準備好新的查詢 |
| -> | 等待多行查詢的下一行 |
| '> | '等待下一行，等待以單引號 ( )開頭的字串的完成 |
| "> | "等待下一行，等待以雙引號 ( )開頭的字串完成 |
| `> | `等待下一行，等待以反引號 ( )開頭的識別碼的完成 |
| /\*> | 等待下一行，等待以開頭的註解完成/\* |

上方表格為MYSQL官方提供

舉個例子比較快理解，如果今天在 MySQL 中，編寫由'或 "字元包圍的字串（例如， 'hello'or "goodbye"），並且 mysql允許您輸入跨多行的字串。當您看到'>或其他如”>，表示您已輸入’或者”等其他符號"，但在字串結束時候尚未輸入’或”。這通常表示您無意中遺漏了引號字元。



本章節結束前複習一下

* 文字輸入不分大小寫。
* 查詢可以多筆資料一起查詢。
* 查詢時，如果用分號結尾代表語句結束，輸入下一筆在用分號結尾，結果會上下顯示。
* 查詢時，如果用逗號結尾代表語句結束，輸入下一筆在用分號結尾，結果會並排顯示。
* 如果不按分號結束，MYSQL惠要求繼續輸入指示，如果想結束語句可以直接\C。
* >前面符號表式，與句當中無意中遺漏了引號字元。

※在MYSQL任何指令結尾一定要有分號;，因為MYSQL指認分號作為語句結束，除了部分指令例外，例如：QUIT。

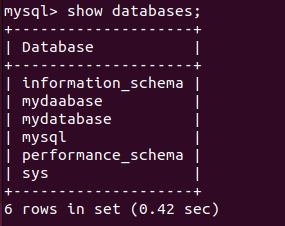
第三章 建立並使用資料庫

基本查詢說完了，接者直接進入如何建立並使用資料庫，首先建立一個觀念，我們使用的是資料庫建立資料表，不是辦公室軟體建立資料，如：微軟EXECL，所以我們必須要有一個資料庫，把它想成一個資料夾，裡面放著資料表。

第一節 找出目前的資料庫

我們首先找出目前伺服器上擁有的資料庫(DATAVASES)，輸入以下指令便能查詢

**SHOW DATABASES;**



現在我們先選擇某一個資料庫，例如我想選擇mydatabase這資料庫，直接再命令窗輸入：

**USE mydatabase;**



另外，USE這指令再結尾可以不用分號作為結尾，現在已經可以使用mydatabase了，但是該資料庫內容皆可以被有權存取該資料庫的其他人修改刪除。因此，您可以向 MySQL 管理員要求使用您自己的資料庫的權限。假設您想調用您的menagerie. 管理員需要執行這句語法：

**GRANT ALL ON menagerie.\* TO 'your\_mysql\_name'@'your\_client\_host';**

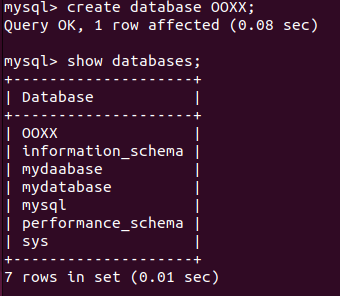
your\_mysql\_name是您的 MySQL 使用者名，your\_client\_host是您連接到伺服器的主機。

第二節 建立資料庫與刪除資料庫

現在我們來建立一個資料庫來玩玩吧，再視窗直接執行下列語句：

**CREATE DATABASE OOXX;**

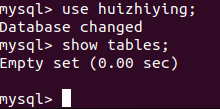
注意這OOXX是隨便取的，實際情形還是必須依照資料庫用途而設立稱，例如：想建立一個灰之影傭兵團的資料表，我們名稱最好直接取名為HuIzhIyIng，這樣一來一目了然。



現在我們選擇了，HuIzhIyIng這剛剛新建立的資料庫，要查看資料去的名稱，請執行此語法

**SHOW TABLES;**

內容自然是空的，因為還沒建立任何資料表。



既然沒有內容那我們就新增內容吧，首先我們必須先執行資料表名稱，只要執行下列語句

**CREATE TABLE shadow\_members (**

**member\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**username VARCHAR(50) NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**password VARCHAR(255) NOT NULL,**

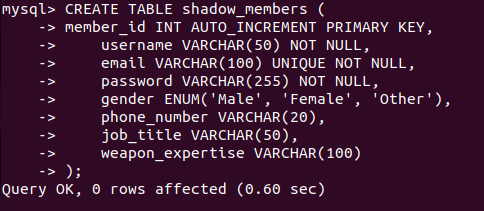
**gender ENUM('Male', 'Female', 'Other'),**

**phone\_number VARCHAR(20),**

**job\_title VARCHAR(50),**

**weapon\_expertise VARCHAR(100)**

**);**



然後輸入資料表的欄位名稱

member\_id INT AUTO\_INCREMENT PRIMARY KEY：

member\_id 是資料表的主鍵（Primary Key），用來唯一標識資料表中的每一筆資料。

INT 表示這是一個整數型態的欄位。

AUTO\_INCREMENT 表示在插入新資料時，member\_id 將自動遞增，每次遞增 1。

PRIMARY KEY 表示 member\_id 是主鍵。

username VARCHAR(50) NOT NULL：

username 是一個字串型態的欄位，最大長度為 50 個字元。

NOT NULL 表示這個欄位不能為空值。

email VARCHAR(100) UNIQUE NOT NULL：

email 是一個字串型態的欄位，最大長度為 100 個字元。

UNIQUE 表示這個欄位的值必須是唯一的，不允許有重複的 email。

NOT NULL 表示這個欄位不能為空值。

password VARCHAR(255) NOT NULL：

password 是一個字串型態的欄位，最大長度為 255 個字元。

NOT NULL 表示這個欄位不能為空值。

gender ENUM('Male', 'Female', 'Other')：

gender 是一個 ENUM（列舉）型態的欄位，表示性別，只能是 'Male', 'Female', 或 'Other' 這三者之一。

phone\_number VARCHAR(20)：

phone\_number 是一個字串型態的欄位，最大長度為 20 個字元。

job\_title VARCHAR(50)：

job\_title 是一個字串型態的欄位，最大長度為 50 個字元。

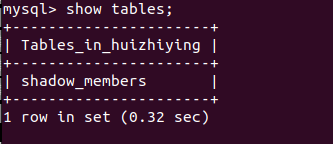
weapon\_expertise VARCHAR(100)：

weapon\_expertise 是一個字串型態的欄位，最大長度為 100 個字元。

可以想像成，我現在有一張檔名shadow\_members的資料表，現在表內已經有我們要輸入的欄位名稱

| member\_id | username | email | password | gender ENUM | **job\_title** | phone\_number | weapon\_expertise |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **,** |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

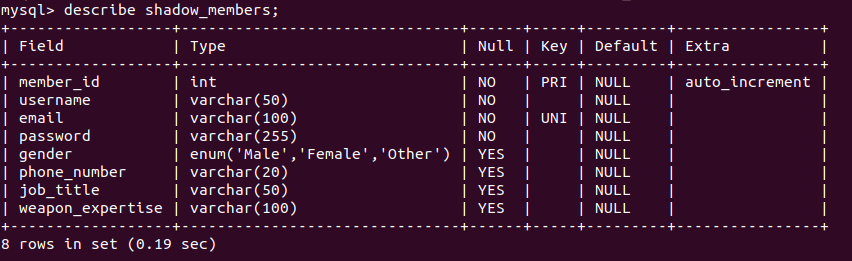
如果建置完成，我們再執行一次指令



發先已經出現一個**shadow\_members資料表**

若要驗證您的資料表是否已經按照您設計的方式創建，請使用以下這段語法

**DESCRIBE shadow\_members;**



現在我們選擇shadow\_members資料表，按照資料表的屬性欄位來建立資料，如以下執行之語法：

**INSERT INTO shadow\_members (username, email, password, gender, job\_title, weapon\_expertise)**

**VALUES**

**('路西安．卡爾玆', 'lucian@example.com', 'password1', 'Male', '實習剣士', '長劍'),**

**('波里斯．貞奈曼', 'boris@example.com', 'password2', 'Male', '王牌剣士', '長劍'),**

**('麥克斯明．里伯克列', 'maximin@example.com', 'password3', 'Male', '智將', '長劍'),**

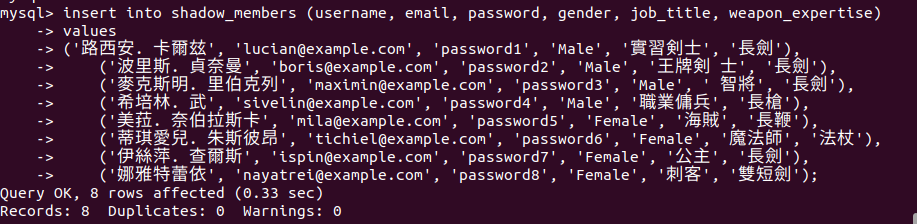
**('希培林．武', 'sivelin@example.com', 'password4', 'Male', '職業傭兵', '長槍'),**

**('美菈．奈伯拉斯卡', 'mila@example.com', 'password5', 'Female', '海賊', '長鞭'),**

**('蒂琪愛兒．朱斯彼昂', 'tichiel@example.com', 'password6', 'Female', '魔法師', '法杖'),**

**('伊絲萍．查爾斯', 'ispin@example.com', 'password7', 'Female', '公主', '長劍'),**

**('娜雅特蕾依', 'nayatrei@example.com', 'password8', 'Female', '刺客', '雙短劍');**



上表實際上少了phone\_number，因為等等我們要學習如何更新資料表

INSERT INTO shadow\_members: 這是 SQL 插入語句的開始，指定將要插入資料的目標資料表為 shadow\_members。

(username, email, password, gender, job\_title, weapon\_expertise): 這部分指定了將要插入資料的欄位順序，這些欄位的順序應該與後面 VALUES 中的值對應。

VALUES: 這是插入資料的關鍵字，它之後會跟隨實際要插入的數值。

每一行的數值：每一行都是一條資料記錄，包含了欄位的具體數值。例如：

第一行插入了 '路西安．卡爾玆' 這個使用者的資料。

第二行插入了 '波里斯．貞奈曼' 這個使用者的資料。

以此類推，每一行都對應一位成員的資訊。

現在shadow\_members資料表已經如同下方表格所顯示

| member\_id | username | email | password | gender ENUM | phone\_number | job\_title, | weapon\_expertise |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 路西安．卡爾玆 | lucian@example.com | password1 | Male | null | 實習剣士 | 長劍 |
| 2 | 波里斯．貞奈曼 | 'boris@example.com | password2' | Male | null | 王牌剣士 | 長劍 |
|  | .  .  .  . | | | | | | |
| 7 | 伊絲萍．查爾斯 | ispin@example.com | password7 | Female | null | 公主 | 長劍 |
| 8 | 娜雅特蕾依 | nayatrei@example.com | password8 | Female | null | 刺客 | 雙短劍 |

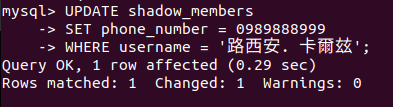
第四節 新增或更新資料表內容

剛剛我們沒有把所有人的電話輸入進去，所以下再要把所有人資料輸入進去，語法如下

**UPDATE shadow\_members**

**SET phone\_number = 0989888999**

**WHERE username = '路西安．卡爾玆';**



現在我查詢看看是不是已經新增上去了，執行下面語法即可

**SELECT username, phone\_number FROM shadow\_members;**

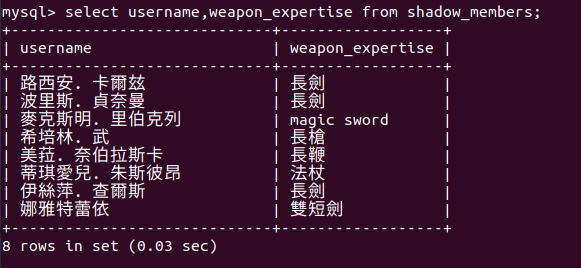


如上表顯示已經把路西安的phone\_number新增上去了，當然我們也可以利用這與法直接修正資料表內容例如我想修改麥克斯明的武器，可以直接執行下列語句

**UPDATE shadow\_members**

**SET weapon\_expertise = magic sword**

**WHERE username = '麥克斯明. 里伯克列';**



如果我想查詢拿長劍當作武器的成員，可以執行下列語句

第五節 查詢資料表

現在我們來查詢我們建立的資料表內容吧，首先示範一個最簡單的查詢資料表內容

**SELECT \* FROM shadow\_members;**



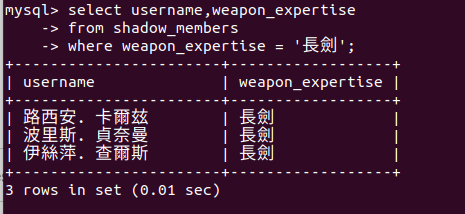
SELECT \*: 表示要選擇所有的欄位。

FROM pet: 表示要從名為 shadow\_members 的資料表中進行查詢。

**SELECT username,weapon\_experise**

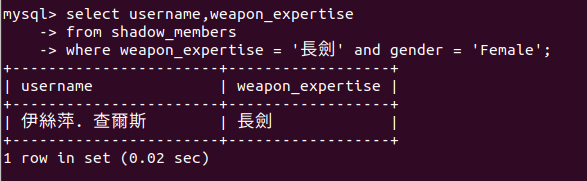
**FROM shadow\_members**

**WHERE weapon\_experise = ‘長劍’;**



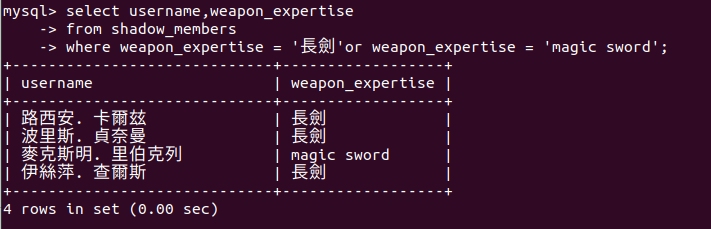
執行結果是路西安，波里斯及伊斯萍。

我們也可以在語句中用and 加上限制，例如我們在後面加上and dender = ‘female’



執行結果只有伊斯萍。

除了and之後我們也可以利用 or 來指定結果，例如在**WHERE weapon\_experise = ‘長劍’後面加上 or weapon\_experise = ‘magic sword’;**



執行結果是路西安，波里斯、伊斯萍與麥克斯明。

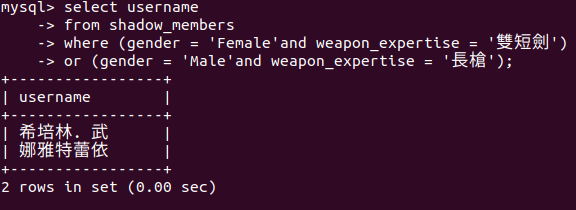
And 與 or 的指令也可以混合使用，不過最好使用括號來明確指示條件應如何將條件分組，指令如下

**SELECT username**

**FROM shadow\_members**

**WHERE (gender = ‘female’ and weapon\_experise = ‘雙短劍’)**

**or(gender = ‘Male’ and weapon\_experise = ‘長槍’)**



執行結果是希培林與娜亞。

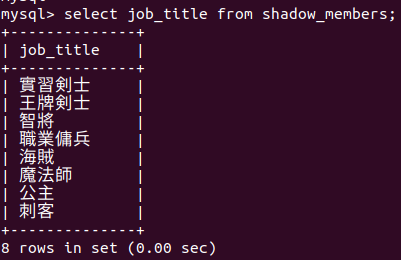
今天我如果只想查詢所有人的密碼，只需要在**SELECT username後面加上,password，與法如下**

**SELECT username,password FROM shadow\_members**



如果只想查出職業名稱可以執行下列與法。

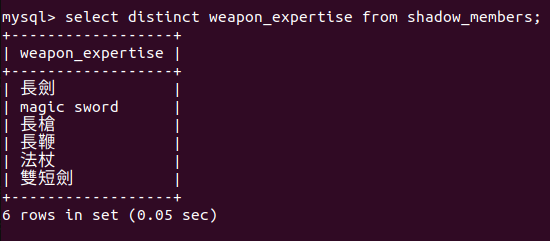
**SELECT job\_title FROM shadow\_members;**



這與指令只查詢單一欄位名稱，所以把job\_title換成其他欄位名稱也能得到相同結果。

我們現在知道有些人的武器有重複出現，如果我只想查詢資料表出現的武器種類，我們可以，執行下面語法

**SELECT DISTINCT weapon\_expertise FROM shadow\_members;**



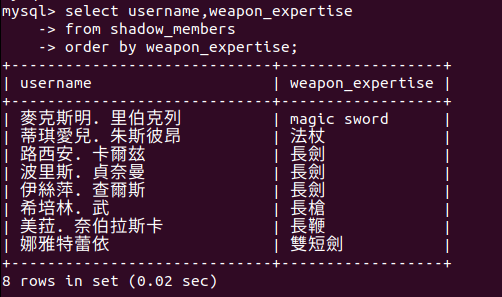
現在我們可以看到只顯示武器種類，重複的值都被挑掉了。

現在來介紹其他查詢方式，我們再利用execl等辦公室軟體在查詢資料的時候，一定都會做排序動作，例如由小至大的方式呈現，只需利用order by，執行與法如下

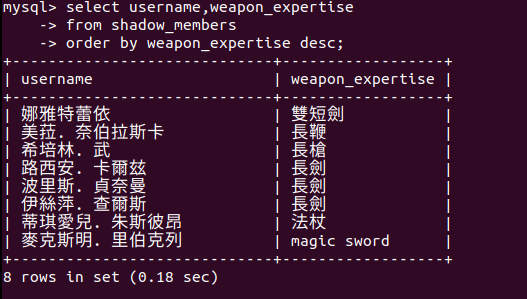
**SELECT username, weapon\_expertise**

**FROM shadow\_members**

**ORDER BY weapon\_expertise;**



如果我們想要倒過來由大至小那我們只要在與法後面加上desc即可



我們也可以按照不同方向進行排序查詢，執行語法如下

**SELECT username, email, weapon\_expertise FROM pet**

**ORDER BY email, weapon\_expertise DESC;**

不一定要使用email，因為本資料表成員姓名皆是中文名，因此用username，較不能看出結果

##### 

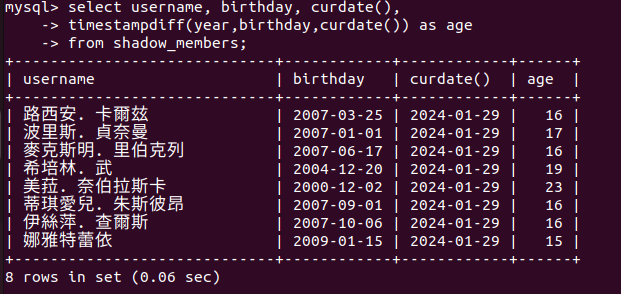
第六節 時間查詢計算

今天我們想計算每個成員他們的年紀，可以使用TIMESTAMPDIFF這指令來完成，CURDATE表示今天日期，也就是說這語法是說我們查詢username, birthday即沒有在資料表的CURDATE()欄位，接著TIMESTAMPDIFF(YEAR,birthday,CURDATE()) AS age來計算它的生日到今天為此用年來顯示結果，並多出一個age的欄位顯示年齡。

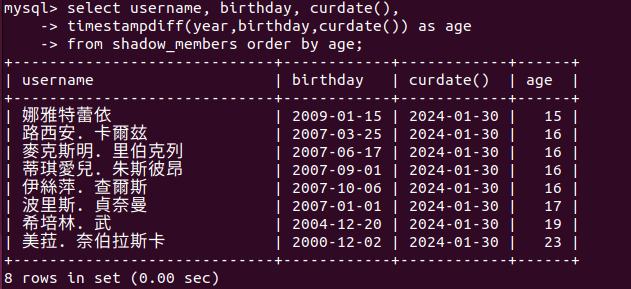
**SELECT username, birthday, CURDATE(),**

**TIMESTAMPDIFF(YEAR,birthday,CURDATE()) AS age**

**FROMshadow\_members;**



如果我們想對進行排序，我們也能在尾句加上ORDER BY birthday，進行A~Z由上而下的排序。

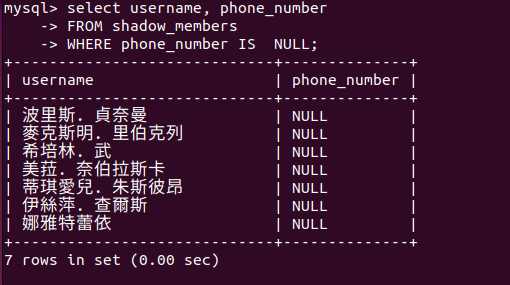


相信大家建立資料表時一定會有一些空值存在，如果想針對某個屬性欄位進行搜索，可以執行下面語法：

**SELECT username, phone\_number**

**FROM shadow\_members**

**WHERE phone\_numer IS NULL;**



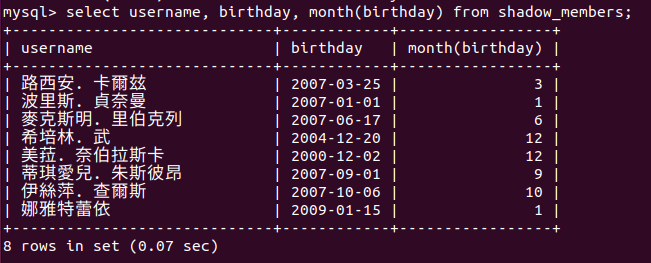
當然如果想搜索非空值的也是可以的，只要在把WHERE phone\_numer IS NULL;修改成WHERE phone\_numer IS NOT NULL;就可以了。

相信大家在日常生活當中，都會問某人幾月出生，今天我們也可以讓mysql回答我們這問題，只是執行下面語法

**SELECT username, birthday,**

**Month(birthday)**

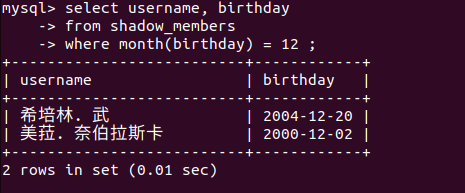
**FROM shadow\_members;**



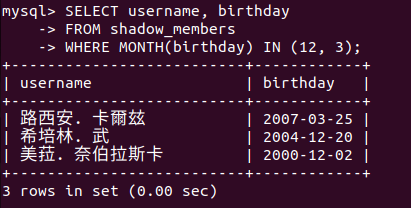
當然不只是月份，年和日也都可以只要把月Month換成year 或 day即可。

那我不想一次看全部，我只想看某個月分，加上

WHERE MONTH(birthday) = 12;



想查詢特定幾個月份執行WHERE MONTH(birthday) in (12,3);即可

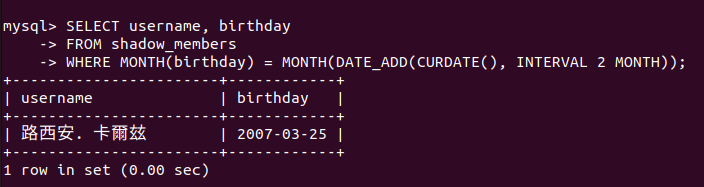


今天如果我想知道下個月或下下個月有誰生日，可以執行下列語法

**SELECT username, birthday**

**FROM shadow\_members**

**WHERE MONTH(birthday) = MONTH(DATE\_ADD(CURDATE(), INTERVAL 2 MONTH));**



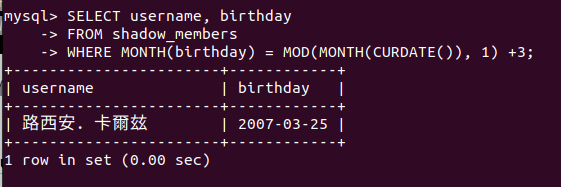
WHERE MONTH(birthday) = MONTH(DATE\_ADD(CURDATE(), INTERVAL 2 MONTH));，語法當中的2代表著是下兩個月後，因為成員表2月沒人生日，輸出結果會是空值，如果想查下下下個月，那直接改成3，以此類推。

今天如果我想知道某一個月的下個月或下下個月有誰生日，可以執行下列語法

**SELECT username, birthday**

**FROM shadow\_members**

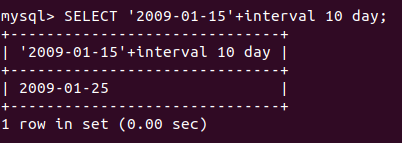
**WHERE MONTH(birthday) = MOD(MONTH(CURDATE()), 1) + 3;**



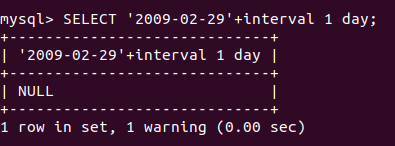
範例中我們選1月份往後推3個月(包含1月)，故得到結果是3月份出生的路西安。

如果只想計算某月份的幾天之後，我們可以執行下列語法

**SELECT ‘2009-01-15’+interval 10 day ;**



但如果我們輸入SELECT ‘2009-02-29’+interval 1 day ;會出現空值，因為2009-02-30不存在。



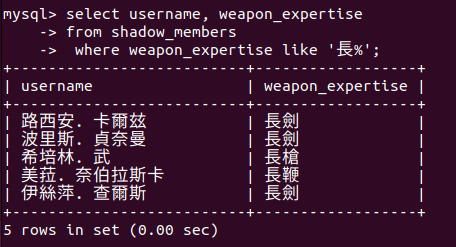
第七節 模式比對

現在我們連學學模式比對搜尋，我們先簡單的來查詢隊員使用的武器吧，請執行下列語法

SELECT username , weapon\_experise

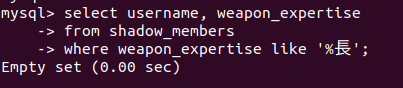
FROM shadow\_members

WHERE ; weapon\_experise like ‘長%’;

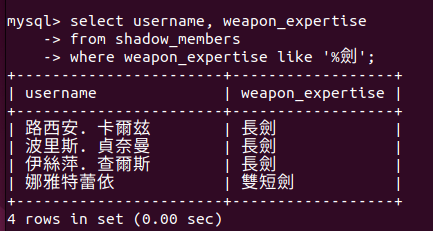


模式比對語法其實就是在輸出結果後面加上like即可，但是%的位子是有分別的

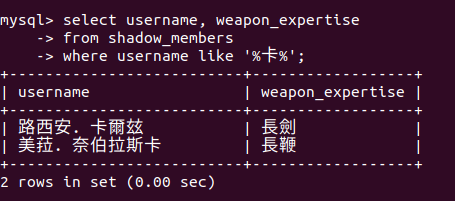
比如說將長放在%後方，結果會如下圖



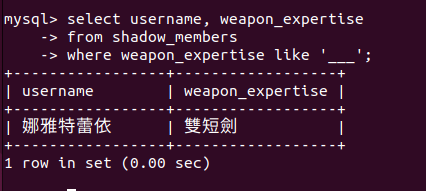
為甚麼呢?因為weapon\_experise的資料當中，字尾並沒有長這個字，如果說換上劍這字，結果後出現下圖所示



現在若不想選擇字首與字尾，只希望有相同的字，呈現出來，很簡單只要用%%把字包裝起來即可，如下圖所表示



如果覺得字太麻煩了，只想要呈現指定字元就好，那麼就要使用\_字符，如下圖所示



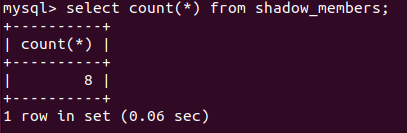
上圖使用\_\_\_三個底線字符，所以結果就是雙短劍。

第八節 計數行

如果今天我們想知道資料表中有幾行資料，我們可以執行

SELECT COUNT(\*) FROM shadow\_wembers;

**SELECT COUNT(\*) FROM shadow\_wembers;**

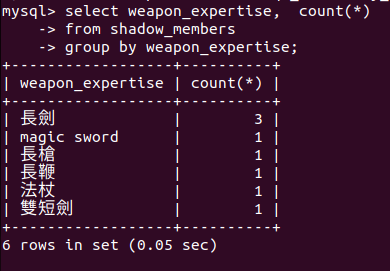


接著我們來計算武器持有者有多人人，我們可以執行下列語句

**SELECT weapon\_experise, COUNT(\*)**

**FROM shadow\_wembers**

**Group by weapon\_experise**

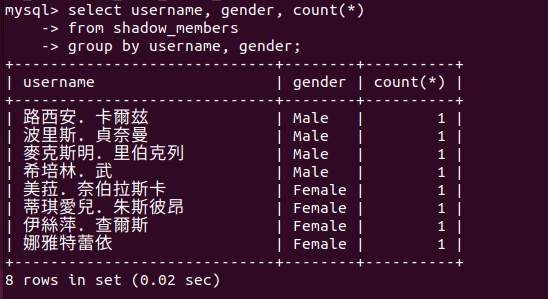


不僅如此，我們還可以加入不同欄位，例如我想知道資料表中成員的性別，可以執行下列語句

**SELECT username, gender, COUNT(\*)**

**FROM shadow\_wembers**

**Group by username, gender;**



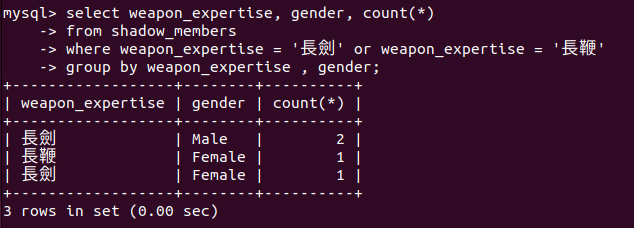
不過查詢成員名性別數量似乎沒太大意義，因為人名都是獨一名稱，不會有重複值出現，這次我們改成查詢武器手持者的性別吧。

**SELECT weapon\_expertise, gender, COUNT(\*)**

**FROM shadow\_wembers**

**WHERE weapon\_expertise=’長劍’ OR weapon\_expertise=’長鞭’**

**Group by weapon\_expertise, gender ;**



這次就可以看的出來成員中拿長劍的有2位男性和1位女性。

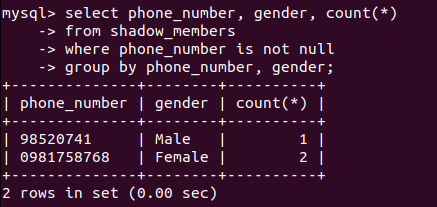
資料表當中一定有空值，如果我們要把空值拿掉了，只要執行下列語法即可

**SELECT phone\_number, gender, COUNT(\*)**

**FROM shadow\_wembers**

**WHERE phone\_number is not null**

**Group by phone\_number, gender ;**



如結果所示，有留手機號碼的有2位女性與1名男性。

第三章 建立關聯表

我們先建立一張成員的寵物表吧

與法如下：

**CREATE TABLE pet (**

**pet\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**pet\_name VARCHAR(50) NOT NULL,**

**pet\_species VARCHAR(50) NOT NULL,**

**pet\_birthday DATE NOT NULL,**

**pet\_type VARCHAR(50) NOT NULL,**

**owner\_username VARCHAR(50) NOT NULL,**

**FOREIGN KEY (owner\_username) REFERENCES shadow\_members(username)**

**);**

CREATE TABLE pet (: 創建一個新的表，名稱為 pet。

pet\_id INT AUTO\_INCREMENT PRIMARY KEY,: 定義一個整數型的列 pet\_id，該列是自動增量的（使用 AUTO\_INCREMENT 關鍵字），並且被指定為主鍵（PRIMARY KEY）。

pet\_name VARCHAR(50) NOT NULL,: 定義一個最大長度為 50 字元的字串列 pet\_name，並且不能為空（NOT NULL）。

pet\_species VARCHAR(50) NOT NULL,: 定義一個最大長度為 50 字元的字串列 pet\_species，並且不能為空。

pet\_birthday DATE NOT NULL,: 定義一個日期型的列 pet\_birthday，並且不能為空。

pet\_type VARCHAR(50) NOT NULL,: 定義一個最大長度為 50 字元的字串列 pet\_type，並且不能為空。

owner\_username VARCHAR(50) NOT NULL,: 定義一個最大長度為 50 字元的字串列 owner\_username，用於存儲寵物的擁有者（主人）的使用者名稱，並且不能為空。

FOREIGN KEY (owner\_username): 這部分指定了外鍵約束的開始，它告訴資料庫這個約束是應用在 owner\_username 列上的。

REFERENCES shadow\_members(username): 這部分指定了外鍵引用的目標，也就是 owner\_username 列的值必須對應到 shadow\_members 表中的 username 列。換句話說，pet 表中的每個 owner\_username 的值都必須存在於 shadow\_members 表的 username 列中，否則數據庫會阻止這樣的關聯。這種約束確保了 pet 表中的每個寵物都有一位有效的擁有者，並且這個擁有者的資訊可以在 shadow\_members 表中找到。

**插入寵物資料**

**INSERT INTO pet (pet\_name, pet\_species, pet\_birthday, pet\_type, owner\_username)**

**VALUES**

**('yellow', '龍', '2017-05-05', '幻獸種', '路西安．卡爾玆'),**

**('blue', '老鷹', '2017-05-05', '現代種', '波里斯．貞奈曼'),**

**('wirte', '蛇', '2018-06-05', '古代種', '美菈．奈伯拉斯卡'),**

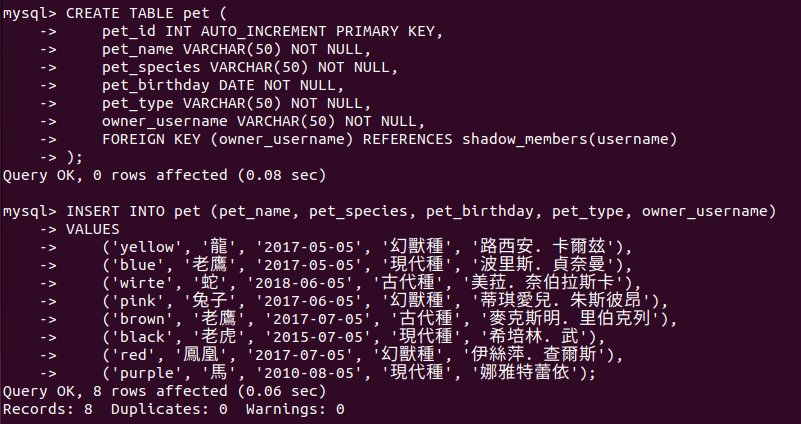
**('pink', '兔子', '2017-06-05', '幻獸種', '蒂琪愛兒．朱斯彼昂'),**

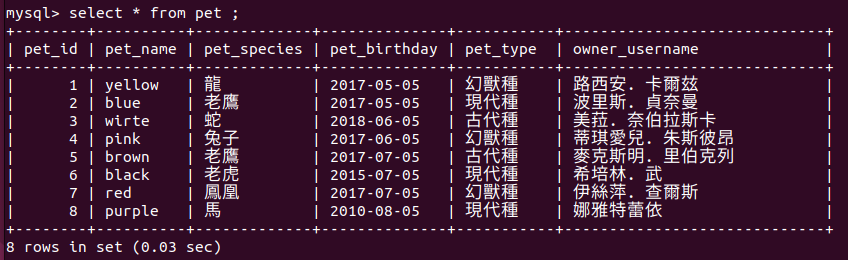
**('brown', '老鷹', '2017-07-05', '古代種', '麥克斯明．里伯克列'),**

**('black', '老虎', '2015-07-05', '現代種', '希培林．武'),**

**('red', '鳳凰', '2017-07-05', '幻獸種', '伊絲萍．查爾斯'),**

**('purple', '馬', '2010-08-05', '現代種', '娜雅特蕾依');**



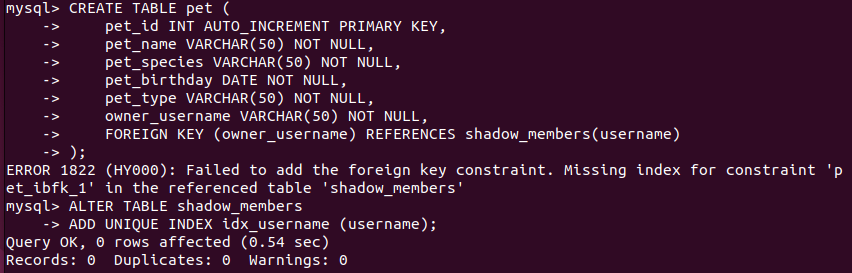


如果出現以下錯誤訊息代表建立外鍵（Foreign Key）時，必須確保參考的表格中有被引用欄位的索引。根據錯誤訊息， shadow\_members 表格中的 username 欄位缺少索引。 為 shadow\_members 表格的 username 欄位添加索引

ALTER TABLE shadow\_members

ADD UNIQUE INDEX idx\_username (username);

使用 ALTER TABLE 來為 shadow\_members 表格的 username 欄位添加一個唯一索引。這樣你再次嘗試建立 pet 表格的外鍵時，就不會再出現 ERROR 1822 的錯誤。



首現我們來學一下，如何利用兩張呈現訊息

**SELECT shadow\_members.username,**

**TIMESTAMPDIFF(YEAR, shadow\_members.birthday, CURDATE()) AS age, pet.pet\_species**

**FROM shadow\_members INNER JOIN pet ON shadow\_members.username = pet.owner\_username**

**WHERE pet.pet\_type ='幻獸種' ;**

恩…..好長啊!，讓我們來好好解釋一下每段語句的意思吧

SELECT shadow\_members.username: 從 shadow\_members 表中選擇 username 列。

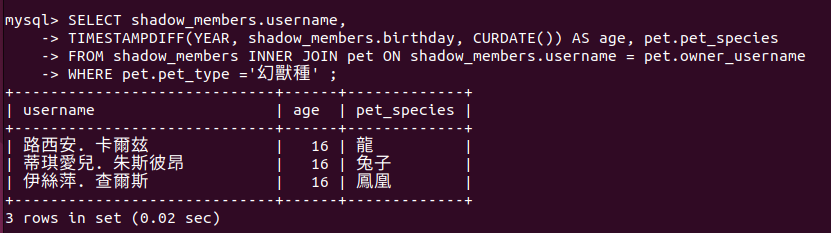
TIMESTAMPDIFF(YEAR, shadow\_members.birthday, CURDATE()) AS age: 計算 shadow\_members 表中每位成員的年齡，使用 TIMESTAMPDIFF 函數來計算生日到當前日期的年數，並且將結果別名為 age。

pet.pet\_species: 從 pet 表中選擇 pet\_species 列。

FROM shadow\_members: 指定查詢的主要表為 shadow\_members。

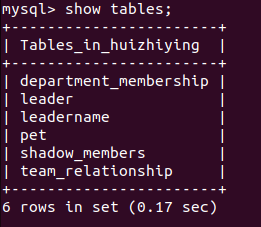
INNER JOIN pet ON shadow\_members.username = pet.owner\_username: 使用 INNER JOIN 關鍵字將 shadow\_members 表和 pet 表結合，條件是 shadow\_members 表中的 username 等於 pet 表中的 owner\_username。

WHERE pet.pet\_type ='幻獸種': 篩選只有那些 pet\_type 列為 "幻獸種" 的寵物。

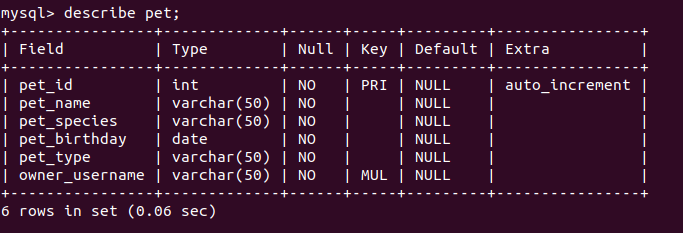


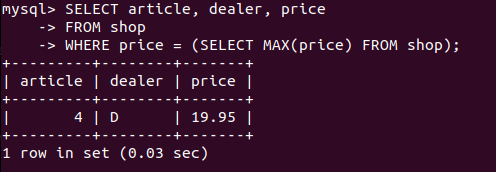
第四章 取得有關資料庫和表格的信息

查詢目前資料庫當中有幾張資料表



查詢資料表欄位的設定



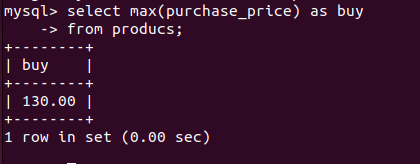


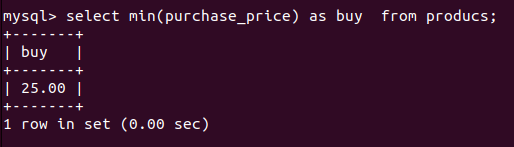
第六章 用聚合函數查詢

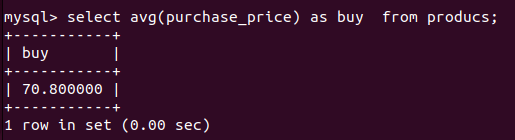
聚合函數

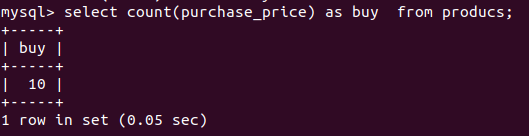
|  |  |
| --- | --- |
| 函數 | 說明 |
| Avg | 計算平均數 |
| Max | 計算最大值 |
| Min | 計算最小值 |
| Sum | 計算總和 |
| Count | 計算資料筆數 |

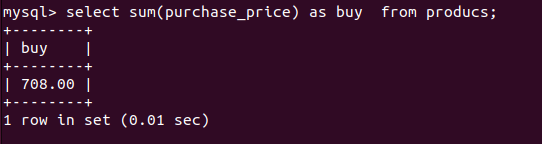
我們先簡單使用這5個聚合函數吧



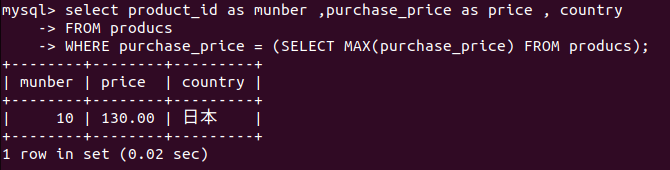








相信大家看到顯示結果，都知道聚合函數是甚麼了吧，那我們還複雜一點的使用聚合函數吧



第七章 其他查詢範例

這次我們直接拿官方文本來直接操作，學習其他查詢方式吧，首先同樣的我們先把資料表建立起來

**CREATE TABLE shop (**

**article INT UNSIGNED DEFAULT '0000' NOT NULL,**

**dealer CHAR(20) DEFAULT '' NOT NULL,**

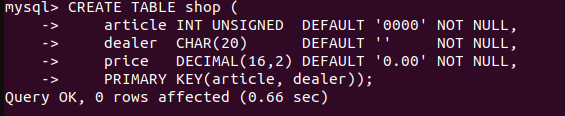
**price DECIMAL(16,2) DEFAULT '0.00' NOT NULL,**

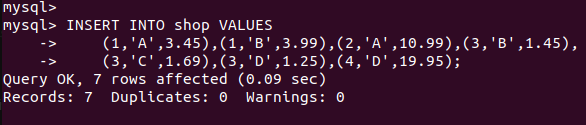
**PRIMARY KEY(article, dealer));**

**INSERT INTO shop VALUES**

**(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),**

**(3,'C',1.69),(3,'D',1.25),(4,'D',19.95);**





我們來稍微解釋一下，語法意思吧

CREATE TABLE shop(：先建立一張shop的資料表

article：屬性欄位是一個無符號的整數，預設值為 '0000'，不可為空值。

dealer：屬性欄位是最大長度為20的字串，不可為空值。

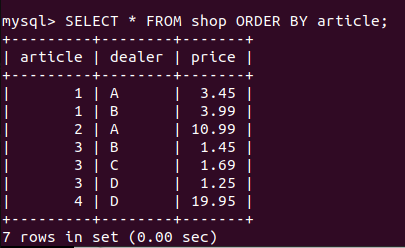
price：屬性欄位是一個十進制數字，總共有16位數字，小數點後有2位，預設值為 '0.00'不可為空值。

資料表有一個主鍵，主鍵由 article 和 dealer 這兩個欄位的組合構成，確保了每一筆資料的組合都是唯一的。

下一步，透過INSERT INTO語句，將範例的資料插入shop資料表內。(1,'A',3.45)，1代表article欄位，A代表dealer欄位，最後3.45代表price欄位。以此類推。

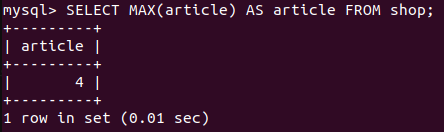
現在我們查詢剛剛建立的資料表吧，我們使用SELECT \* FROM shop ORDER BY article;，我們結合ORDER BY 來排序查詢資料表，結果如下圖所示

SELECT \* FROM shop ORDER BY article;



如果我想知道最高的商品編號是多少，那我們必須使用聚合函數的MAX，語法如下圖所示

SELECT MAX(article) AS article FROM shop;

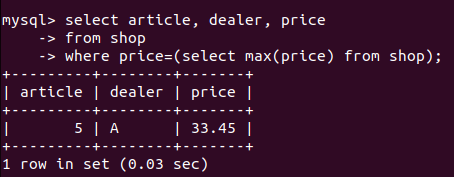


現在我要找出最貴物品的編號、經銷商和價格，語法如下。

SELECT article, dealer, price

FROM shop

WHERE price=(SELECT MAX(price) FROM shop);



本次結果，我們來講解(SELECT MAX(price) FROM shop)，這段語句，這語句是SQL的子查詢（Subquery），首先我們先解釋主查詢，先前我們所做的查詢接是主查詢，也就是說

SELECT \* FROM shop 這是我們主要查詢的結果，而主查詢語句當中，又給予SELECT 查詢語句稱子為子查詢。並且可以用來檢索資料，這些資料稍後可能會用於主查詢的條件、選擇列表或其他部分。子查詢允許你在一個查詢的結果中執行另一個查詢，並將內部查詢的結果用於外部查詢。它為 SQL 查詢提供了更大的靈活性，允許根據內部查詢的結果進行更複雜的查詢操作。在這個例子中，(SELECT MAX(price) FROM shop) 就是一個子查詢，它返回 shop 表中 price 列的最大值。這個子查詢的結果然後被用於主查詢的條件，選擇了具有最大價格的行。

我們拆解來看則是

SELECT article, dealer, price

表示我的查詢結果欄位要有SELECT article, dealer, price

FROM shop

資料表名稱是shop

WHERE price=(SELECT MAX(price) FROM shop);

執行的結果是幫我找出price欄位當中數值最大的值

如果不想利用子查詢可以利用以下兩種方法

SELECT s1.article, s1.dealer, s1.price

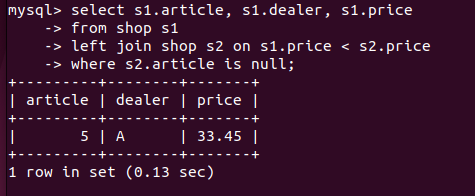
FROM shop s1

LEFT JOIN shop s2 ON s1.price < s2.price

WHERE s2.article IS NULL;

FROM shop s1: 表示查詢的表shop，並使用 s1 作為表的shop表的別名。LEFT JOIN shop s2 ON s1.price < s2.price: 這是一個自身連接（self-join），通過價格進行自身連接。LEFT JOIN 會找出每一行 s1 中的商品，然後在 s2 中找出價格比 s1 高的所有行。WHERE s2.article IS NULL: 這個條件確保只有那些在 s1 表中價格最低的行會被選取。具體而言，如果 s1 的某行沒有比它價格更低的 s2 行，那麼 s2.article 將是 NULL。這個查詢使用了自我聯接（self-join），即將相同表格（shop）視為兩個不同的實例，分別標記為 s1 和 s2。這樣的做法是為了在同一行中比較相同商品的不同價格。在這裡，s1 代表主表格中的每一行，而 s2 代表同一表格中比 s1 擁有更高價格的所有行。這樣的自我聯接允許我們比較同一商品的不同價格，並找到每個商品的最高價格對應的行。

也就是說，我們雖然只有1張shop，但我們可以利用s1與s2的語法，一張當作兩張，現在定義2張shop表，取名s1與s2開始進行比較，輸出的結果是s2中的數值大於s1。



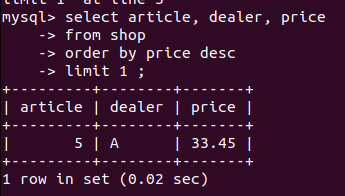
SELECT article, dealer, price

FROM shop

ORDER BY price DESC

LIMIT 1;

透過 ORDER BY price DESC 將商品按照價格降序排序，然後使用 LIMIT 1 只取第一行，即最高價格的商品信息。最終的結果是顯示 shop 表中價格最高的商品的 article、dealer 和 price。



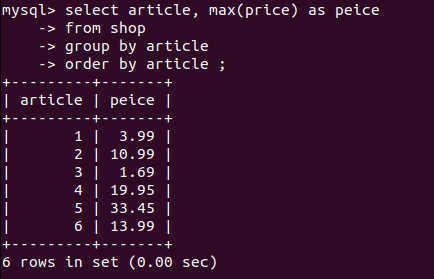
這次來查詢artice編號當中的最大值吧

SELECT article, MAX(price) AS price

FROM shop

GROUP BY article

ORDER BY article;



這個查詢的目的是找出每個 article 對應的最高價格。透過 MAX(price) 函數取得每個 article 的最高價格，同時使用 GROUP BY article 進行分組，確保每個 article 只有一行結果。最後，透過 ORDER BY article 依照 article 的升序排序。

找出price最貴的一個或多個dealer，可以執行以下語法

SELECT article, dealer, price

FROM shop s1

WHERE price=(SELECT MAX(s2.price)

FROM shop s2

WHERE s1.article = s2.article)

ORDER BY article;

語法解釋

FROM shop s1: 從 shop 表格中取別名為 s1 的資料集。

WHERE price=(SELECT MAX(s2.price)

FROM shop s2

WHERE s1.article = s2.article):

目的是找出對應相同 article 中的最高價格。對於每一行 s1，子查詢

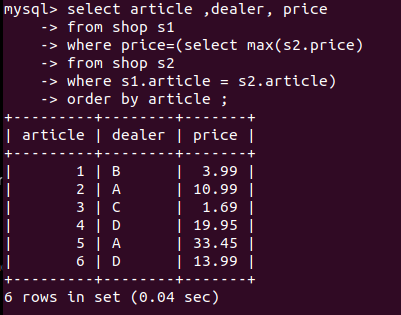
(SELECT MAX(s2.price)

FROM shop s2

WHERE s1.article = s2.article)

會計算相同 article 中的最高價格。整個條件 WHERE price=... 確保只有最高價格的行會符合條件。ORDER BY article: 最後的 ORDER BY 子句確保結果按照 article 升序排序。

總結來說，這個查詢會返回每個 article 對應的最高價格的那一行資料。



SELECT s1.article, dealer, s1.price

FROM shop s1

JOIN (

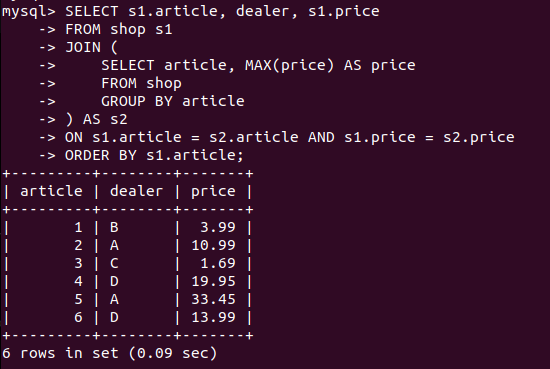
SELECT article, MAX(price) AS price

FROM shop

GROUP BY article) AS s2

ON s1.article = s2.article AND s1.price = s2.price

ORDER BY article;



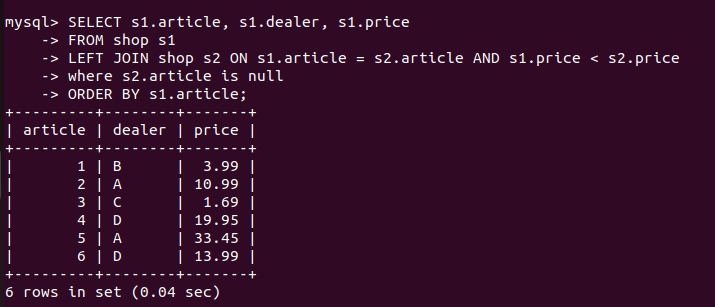
SELECT s1.article, s1.dealer, s1.price

FROM shop s1

LEFT JOIN shop s2 ON s1.article = s2.article AND s1.price < s2.price

WHERE s2.article IS NULL

ORDER BY s1.article;



FROM shop s1: 從 shop 表格中取別名為 s1 的資料集。

LEFT JOIN shop s2 ON s1.article = s2.article AND s1.price < s2.price

這是一個 LEFT JOIN，它匹配相同 article 中價格較高的行。條件 s1.price < s2.price 確保只有價格更高的行會被匹配。WHERE s2.article IS NULL: 這個條件確保只有沒有匹配到的行會被選中，也就是價格最高的行。ORDER BY s1.article: 最後的 ORDER BY 子句確保結果按照 article 升序排序。

**WITH s1 AS (**

**SELECT article, dealer, price,**

**RANK() OVER (PARTITION BY article**

**ORDER BY price DESC**

**) AS `Rank`**

**FROM shop**

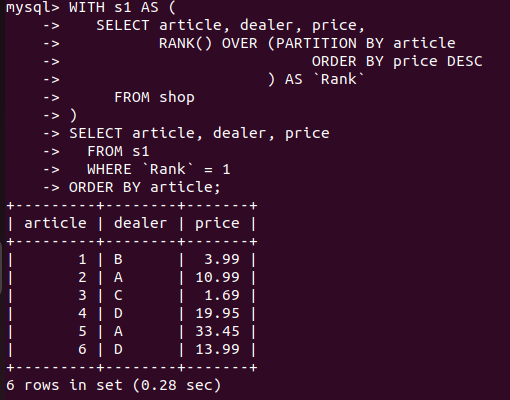
**)**

**SELECT article, dealer, price**

**FROM s1**

**WHERE `Rank` = 1**

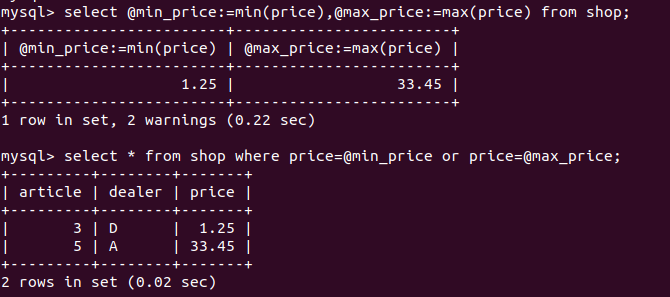
**ORDER BY article;**



WITH s1 AS (...): 定義了一個通用表達式 s1，這個子查詢使用 RANK() 窗口函數，按照 article 分組，按價格降序排列，為每個分組的行賦予排名。

SELECT article, dealer, price FROM s1 WHERE Rank = 1 ORDER BY article;: 在 s1 中選擇排名為 1 的行，這樣就取得了每個 article 對應的價格最高的那一行資料。

現在我們來認識一下，使用者定義變數吧，可以執行以下語法



使用外鍵

我們來操作如何使用使用外鍵吧

首先先建立兩張資料表

**CREATE TABLE parent (**

**id INT NOT NULL,**

**PRIMARY KEY (id)**

**) ENGINE=INNODB;**

**CREATE TABLE child (**

**id INT,**

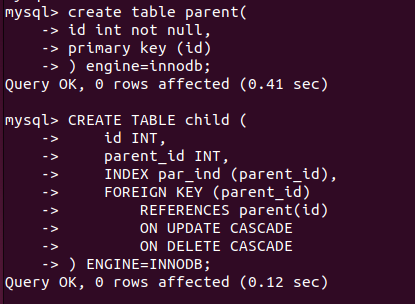
**parent\_id INT,**

**INDEX par\_ind (parent\_id),**

**FOREIGN KEY (parent\_id)**

**REFERENCES parent(id)**

**) ENGINE=INNODB;**



上述 SQL 語句創建了兩個表，一個是 parent 表，另一個是 child 表，並在 child 表上定義了一個外鍵約束，將其與 parent 表的主鍵關聯起來。

解釋：

parent 表:id 是 parent 表的主鍵（Primary Key）。使用 ENGINE=INNODB 定義了 InnoDB 存儲引擎，InnoDB 是 MySQL 中支援事務、外鍵等功能的存儲引擎。在 MySQL 中，ENGINE 用於指定表使用的存儲引擎（Storage Engine）。而 InnoDB 是 MySQL 中的一種存儲引擎，具有支援事務和外鍵等高級功能的特點。InnoDB 存儲引擎解釋如下：InnoDB 是 MySQL 中最常用的存儲引擎之一。你可以使用 BEGIN TRANSACTION、COMMIT 和 ROLLBACK 等 SQL 陳述句來控制事務的開始、提交和回滾。InnoDB 也提供了行級別的鎖定，這有助於防止數據庫的並發問題，多個事務可以同時操作數據，而不會相互干擾。支援外鍵約束：外鍵約束是一種數據庫約束，用於確保兩個表之間的關聯完整性。當你在表中定義外鍵時，你可以設置外鍵約束，這樣插入、更新或刪除數據時就會檢查這種關聯的完整性。InnoDB 存儲引擎支援外鍵約束，這意味著你可以在表中使用 FOREIGN KEY 來定義外鍵，同時通過外鍵約束來確保參照完整性。使用 ENGINE=INNODB 就是告訴 MySQL 使用 InnoDB 存儲引擎來創建這個表。這樣可以充分發揮 InnoDB 存儲引擎的事務支援和外鍵約束等功能，提高數據庫的安全性和完整性。

另外，上述當中提到的存儲引擎（Storage Engine）是指資料庫管理系統（DBMS）中負責處理資料的存儲和檢索的模組或組件。存儲引擎定義了資料庫中數據的儲存結構、查詢處理、事務處理等方面的規則和方法。在一個資料庫中，可以使用不同的存儲引擎來管理不同的表。不同的存儲引擎有不同的特性和用途，可以根據應用的需求來選擇最適合的存儲引擎。常見的存儲引擎包括 InnoDB、MyISAM、Memory（或者叫 Heap）、PostgreSQL 的 PGSQL、SQLite 的 SQLite3 等。

每種存儲引擎都有其優勢和限制，例如，某些引擎可能更適合處理讀取密集型操作，而另一些引擎可能更適合處理寫入密集型操作。在選擇存儲引擎時，需要考慮應用的性能需求、事務支援、故障恢復等因素。

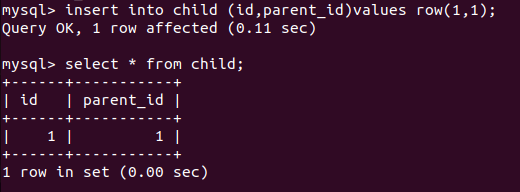
child 表:id INT：在 child 表中定義了一個 id 列，這是一個整數類型的列，用來唯一識別每一行資料。parent\_id INT：定義了一個 parent\_id 列，這是 child 表中與 parent 表關聯的外鍵列。這個列的值將參考 parent 表中的 id 列。INDEX par\_ind (parent\_id)：這是在 parent\_id 列上創建的一個索引，目的是提高查詢效率。索引可以讓資料庫系統更快速地找到特定值，尤其是在使用 WHERE 子句進行查詢時。FOREIGN KEY (parent\_id) REFERENCES parent(id)：定義了一個外鍵約束，將 parent\_id 列設為外鍵，參照的是 parent 表中的 id 列。這表示 child 表的每一行 parent\_id 的值必須存在於 parent 表的 id 列中，確保了兩個表之間的一致性。ENGINE=INNODB：定義了使用 InnoDB 存儲引擎。InnoDB 提供了事務支援和外鍵約束，確保資料的完整性和一致性。

接著parent與child表當中執行下列語句

**INSERT INTO parent (id) VALUES ROW(1);**

**INSERT INTO child (id,parent\_id) VALUES ROW(1,1);**





VALUES 用於指定要插入、更新或選取的值。而 ROW 關鍵字可用於指定一行中的值。在 VALUES ROW(1, 1) 中：ROW 表示接下來的括號內的值應該被視為一行的值。

(1, 1) 表示這一行的值有兩個部分，分別是 1 和 1。簡而言之，VALUES ROW(1, 1) 表示要插入一行資料，並指定該行中各欄位的值。在這個例子中，有兩個欄位，分別是 id 和 parent\_id，它們的值分別為 1 和 1。

