關於Null的介紹

Null，為一種missing、unknown value，發生可能漏填、沒有資料或其他可能因素

- 空值不等同0、或者是""空字串

空值相關的計算，以people table來做以下練習

針對name欄位

```
SELECT COUNT(name)
FROM people;
```

針對birthdate欄位

```
SELECT COUNT(birthdate)
FROM people;
```

針對所有欄位

```
SELECT COUNT(*)
FROM people;
```

condition條件關鍵字IS NULL，可判定該值是否為NULL，若是則保留該筆資料

```
SELECT COUNT(*)
FROM people
WHERE birthdate IS NULL;

SELECT COUNT(*)
FROM people
WHERE name IS NULL;
```

可以測試看看以下這種矛盾的寫法，思考一下

EX1:

```
SELECT COUNT(birthdate)
FROM people
WHERE birthdate IS NULL;
```

EX2:

```
SELECT COUNT(name)
FROM people
WHERE name IS NULL;
```

例題練習1:

從people table中

找出deathdate欄位為空的，還存活的人的name欄位

例題練習2:

從films table中

找出沒有記載budget預算的電影的title欄位

例題練習3:

從films table中

計算，language欄位，沒有記載語言的數量

文字Pattern 比對

以下這種嚴格的比較，需要完全一樣

```
SELECT name
FROM people
WHERE = "Aaron Hann";
```

萬用字元wildcard

之前已經有用過一種萬用字元 *號，代表所有column

EX:

SELECT * 或者是 SELECT COUNT(*)

此觀念源自Regular Expression正規表達式，以下舉一個常見例子

譬如要求註冊會員信箱，希望信箱符合某種格式，也就是上面提到pattern

- 下面例子的pattern即為結尾都是.com，前面則是任意
  - 然後不同家郵件會有各自的pattern

aaa@gmail.com
ccc@gmail.com
ddd@yahoo.com
eee@msn.com

(有興趣可以去查複雜的正規表達式用法，他常見於語言方面的處裡)

SQL中的pattern比對LIKE與NOT LIKE

SQL裡面的pattern比對沒辦法做到太複雜，只能做基本的

有兩種wildcard

1. %，表示任意長度字串，0~n個包含0
2. _，表示任意單一字母

嘗試LIKE，表示符合此pattern的資料

```
SELECT name
FROM people
WHERE name LIKE  'Aaron %';

SELECT name
FROM people
WHERE name LIKE 'Aaron _shmore';
```

嘗試NOT LIKE，表示找出不符合此pattern的資料

```
SELECT name
FROM people
WHERE name NOT LIKE 'Aaron %';
```

例題練習4

從people table中，在name欄位中，找出所有名稱為B開頭的，長度不拘

- 最後秀出欄位name

例題練習5

從people table中，在name欄位中，找出所有名稱第二個字為r，長度不拘

- 最後秀出欄位name

例題練習6

從people table中，在name欄位中，找出所有名稱"不是"A開頭的，長度不拘

- 最後秀出欄位name

Aggregation Functions聚合函數

- 可以將多筆records/rows資料的某欄位的值，聚合成單一值稱做aggregation聚合
  以下提供幾個常用的Aggregation Functions:
  AVG()平均、SUM()加總、MAX()找最大值、MIN()找最小值
  形式: AVG(columnName)

我們利用films table，針對budget來做Aggregation

EX:

算出平均budget

```
SELECT AVG(budget)
FROM films;
```

EX: 為aggregation結果重新命名

名稱，大小寫沒差，都會被轉成小寫

```
SELECT AVG(budget) AS average_budget
FROM films;
```

可一次獲取多個aggregation結果

一次算處AVG、SUM、MIN、MAX，可以仿照上面例子都幫他們取別名

```
SELECT AVG(budget), SUM(budget), MIN(budget), MAX(budget)
FROM films;
```

例題練習7

計算films table的duration欄位的總和

例題練習8

計算films table的duration欄位的平均

例題練習9

計算films table的duration欄位地最大值與最小值(最短/最長片長時間)

例題計算10

統計films table的收入，依據gross欄位

- "這些計算會跳過Null，因此有時候先得知Null數量會比較客觀"

分別想要得到以下統計值

1. total amount grossed by all films
2. avg amount grossed by all films
3. the amount grossed by the worst performing film
4. the amount grossed by the best performing film

配合WHERE針對部分資料做統計Aggregation

譬如我們可以針對2010年以後(含2010)，的全部電影的總預算
以films table來做

```
SELECT SUM(budget)
FROM films
WHERE release_year >= 2010;
```

例題練習11

計算films table中的總收入透過gross欄位
時間設置在2000年以及2000年後

例題練習12

計算films table中，title開頭為A的，電影的總收入gross

例題練習13

計算在1994年films table，收入gross表現最差的值

例題練習14

計算在2000~2012年間(可採用BETWEEN AND關鍵字)
中表現最好的收入gross值，從films table來做

SELECT中允許簡單數值計算

常見的symbol: +加、-減、*乘、/除
這比較少使用，就是一個拿資料庫系統當計算機的概念
(有時候表示一些比例、權重可能會用這個東西)

EX:

```
SELECT (4 * 3);
```

EX:

```
SELECT (4 / 3);
```

若要轉為浮點數(小數)，僅需加上floating point

EX:

```
SELECT (4.0 / 3.0);
```

可以測試一下，下面指令，會發現他的整數除法有些不同

- 平常的程式碼，通常這種情況會自動轉成浮點數
  (表示SQL這方面式相對嚴謹、可預期，較不會有強轉型問題)
  SELECT (10 / 3);

簡單數值計算也可以應用在欄位之間

譬如我們想要獲得一個電影淨利，利用films table來計算

- 利用gross所得 - budget預算
- 最後title、以及計算過後的欄位(gross - budget)

SELECT title, (gross - budget)
FROM films;
但這個有個情況，若數值是null就會導致結果為null
(要注意，不能有數值與字串一起運算，結果不可預期)

- 下面例子，則可保證計算結果都有的情況
  (但是對於效能負擔會較大，有時候我們會在事後程式接收該表單結果的時候再做處理，降低database負擔)
  SELECT title, (gross - budget)
  FROM films
  WHERE (gross - budget) IS NOT NULL;

Aliasing欄位匿名/別名

使用情境:

- 用於aggregation結果的欄位
- 或者欄位名稱太相似也可以取別名
  - database有一種方法是可以把兩個table合併
    (這種時候就高機率會出現欄位名稱一樣)
- 欄位計算後解果也可以做匿名

EX:
延續上面例子淨利配合Aliasing，這樣結果可讀性會比較高
SELECT title, (gross - budget) AS net_profits
FROM films
WHERE (gross - budget) IS NOT NULL;

EX:
aggregation結果配合Aliasing

SELECT MAX(budget) AS max_budget, MAX(duration) AS max_duration
FROM films;

例題練習15

顯示title欄位以及duration播放時間改以小時為單位(採用浮點數)，並且別名取作duration_hours，原本撥放時間為分鐘，因此要去以除法做轉換

- 最終會有兩個欄位title與duration_hours