

# MySQL資料庫系統建置與管理

講師：張明捷

email : [jerrychang@ispan.com.tw](mailto:jerrychang@ispan.com.tw)

# 目錄

---

1. 前言(Preface)
2. MySQL8.0 安裝與設定(Installation & Settings)
3. 基本查詢(Basic Query)
4. 表格(Tables)
5. 空值與預設值(Null & Default)
6. 主鍵(Primary Key)
7. 增刪查改(CRUD)
8. 外來鍵(Foreign Key)
9. 日期與時間(Date & Time)
10. 連結(Join)
11. 分組與聚成(Group By & Aggregation)
12. 排序(Order by)

# 目錄

---

13. 限制與跳過(Limit & Offset)

14. 子查詢(Subquery)

15. 區別(Distinct)

16. 模糊搜尋(Like)

17. 在內與非在內(In & Not In)

18. 關鍵字(Case)

19. 變數(Variables)

20. 預儲程序(Stored Procedure)

21. 交易(Transaction)

22. 資料庫正規化 (Database Normalization)

# 前言(Preface)

# 前言(Preface)

---

在開始學習MySQL資料庫系統建置之前...

## ● SQL是什麼？

- Structured Query Language，是被用於關聯式資料庫管理系統(RDBMS)中的結構化查詢語言，可查詢資料庫(Database)中的資料。
- 1970年代由IBM研究院的Edgar Codd發表，奠定了關聯式資料庫的發展，曾獲得圖靈獎。

## ● SQL的特色是？

- 具有強大的靈活度，在其他程式中可能需要大量的程式碼才能執行的事件，在SQL中可能只需要單獨一句就表達出來了。
- SQL包含了
  1. DDL：負責產生、修改、刪除資料庫物件定義的語言，如CREATE、DROP等。
  2. DML：負責新增、修改、刪除表格資料的處理語言，如INSERT、UPDATE、DELETE等。
  3. DQL：負責查詢資料庫內所含資訊的語言，如SELECT。
  4. DCL：負責處理資料存取權限相關的語言，如GRANT、DENY等。

# 前言(Preface)

---

## ● Database又是什麼？

- 資料庫就是可以存放資料的地方，可能包含了若干個表格(Table)、預儲程序(Stored Procedure)、函數(Function)等。
- 思考一下，表格、資料庫的概念在生活中是不是隨處可見？
- 根據不同的功能，可以在資料庫中新增需要的表格。

## ● 關聯式資料庫管理系統(RDBMS)

- 有關RDBMS：
  1. 可使用通用的互動式查詢語言(SQL)操作
  2. 可同時建立多個表格並儲存資料
  3. 可使用SQL來表示多個表格之間的關係
  4. 可由若干個關聯式資料庫所組成

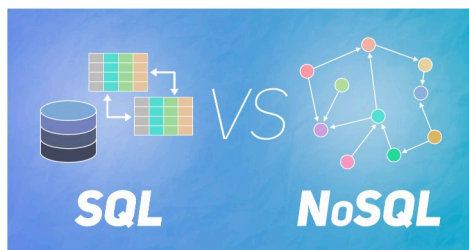
## ● 市面上有哪些RDBMS？

- Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL, Maria DB, SQLite等。

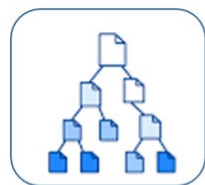
# 前言(Preface)

---

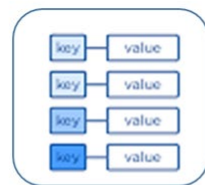
## ● 關聯式資料庫(SQL) vs. 非關聯式資料庫(NoSQL)



### NoSQL



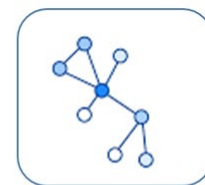
Document Store



Key-Value Store



Wide-Column Store



Graph Store

## ● 市面上有哪些NoSQL database ?

- MongoDB, Redis, Elasticsearch, Cassandra等。

# 前言(Preface)

---

## ● 有關MySQL的歷史，以及後續的演化

- 1995年由瑞典MySQL AB公司開發。
- 2008年被昇陽公司(Sun)收購，2010年甲骨文公司(Oracle)收購昇陽
- 2009年傳出併購案開始，Monty Widenius就著手將MySQL原始碼複製(Fork)出去成為了MariaDB。
- 2013年維基百科(Wikipedia)將其資料從MySQL轉換到MariaDB。
- 其優勢在於開源、低成本、功能性不比商業用資料庫系統但足以應付一般中小型企業或大眾使用者的需求。
- LAMP的開源軟體組合，在2010年以前的資訊界創造了潮流。

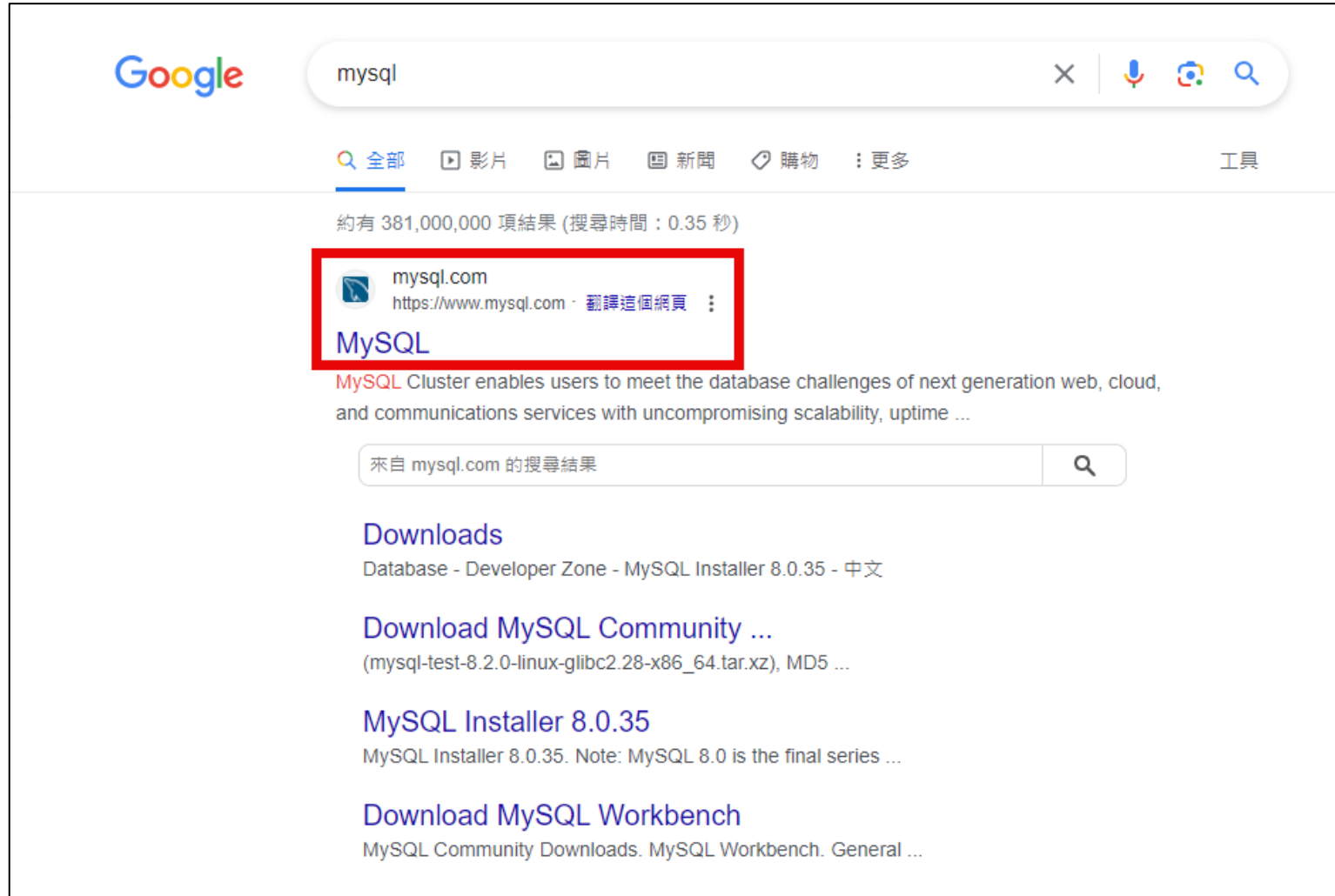
## ● 常見MySQL圖形管理工具

- MySQL Workbench, Navicat, phpMyAdmin

# MySQL8.0安裝與設定 (Installation & Settings)

# MySQL8.0安裝與設定(Installation & Settings)

## ● 下載與安裝



# MySQL8.0安裝與設定(Installation & Settings)



The screenshot shows the MySQL website homepage. The top navigation bar includes the MySQL logo, the tagline "The world's most popular open source database", a search icon, and links for "Contact MySQL", "Login", and "Register". Below this, a secondary navigation bar features "MYSQL.COM", "DOWNLOADS" (highlighted with a red box), "DOCUMENTATION", and "DEVELOPER ZONE". A third navigation bar lists "Products", "Services", "Partners", "Customers", "Why MySQL?", "News & Events", and "How to Buy". The main content area features a large banner for the "MySQL and HeatWave Summit", celebrating MySQL's 30th Anniversary, scheduled for April 22-23, 2025, at the Oracle Conference Center in Redwood Shores, California. A "Free Event - Register Now" button is prominently displayed. Below the banner, four featured products are listed: HeatWave, MySQL Enterprise Edition, MySQL for OEM/ISV, and MySQL Cluster CGE, each with a brief description and a "Learn More" link. A "Start chat" button is visible in the bottom right corner.

MySQL  
The world's most popular open source database

MYSQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE

Products Services Partners Customers Why MySQL? News & Events How to Buy

## MySQL and HeatWave Summit

Keynotes, Learning Sessions, Demos, and Networking  
Celebrate MySQL's 30th Anniversary!

April 22-23, 2025  
Oracle Conference Center - Redwood Shores, California

**Free Event - Register Now**

### HeatWave

Use automated and integrated generative AI and machine learning (ML) in one cloud service for transactions and lakehouse scale analytics. Get faster insights from all your data with unmatched performance and deploy apps in your choice of cloud providers.  
[Learn More »](#)

### MySQL Enterprise Edition

The most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.  
[Learn More »](#)

### MySQL for OEM/ISV

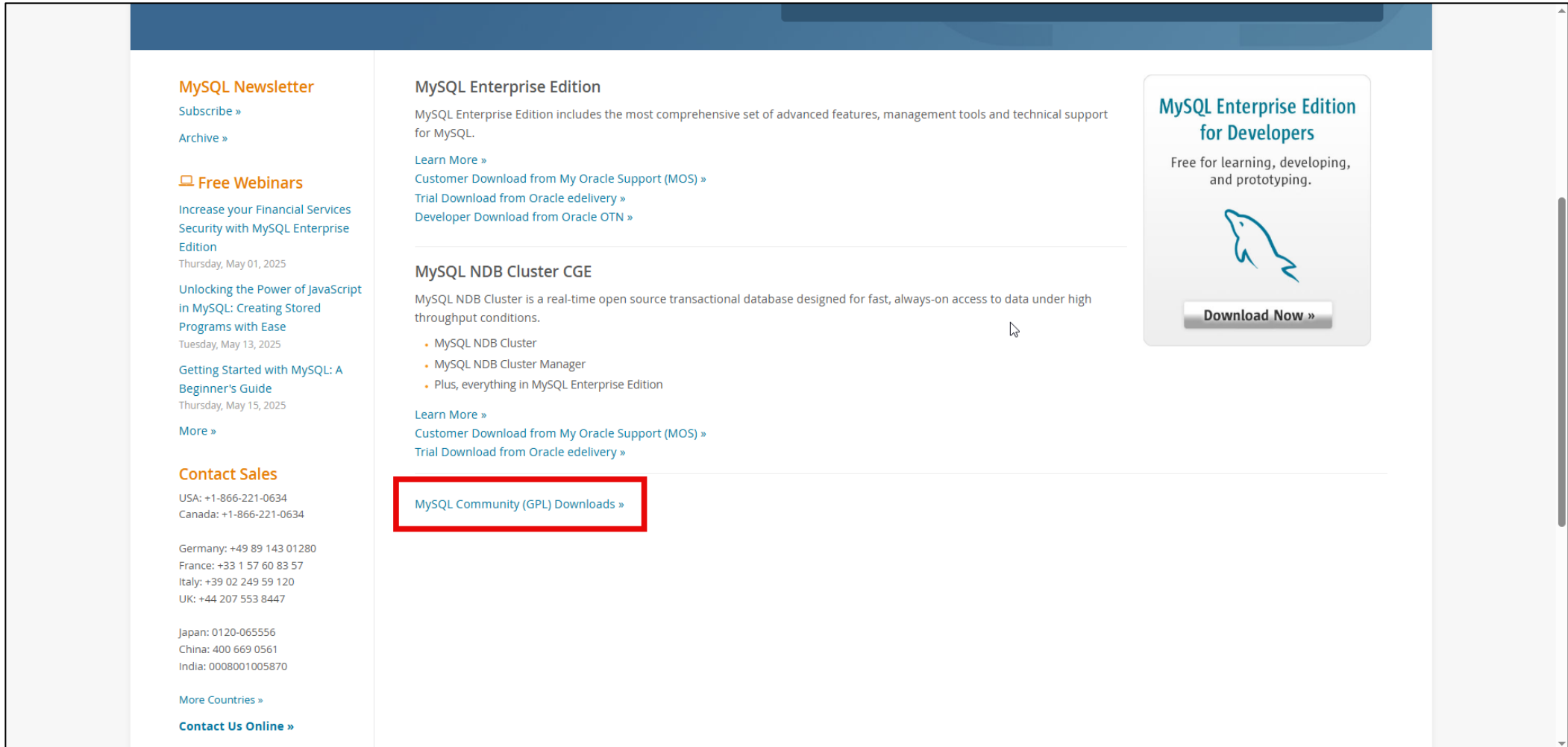
Over 2000 ISVs, OEMs, and VARs rely on MySQL as their products' embedded database to make their applications, hardware and appliances more competitive, bring them to market faster, and lower their cost of goods sold.

### MySQL Cluster CGE

MySQL Cluster enables users to meet the database challenges of next generation web, cloud, and communications services with uncompromising scalability, uptime and agility.  
[Learn More »](#)

[Start chat](#)

# MySQL8.0安裝與設定(Installation & Settings)



The screenshot displays the MySQL website's main content area. On the left, there are sections for the MySQL Newsletter, Free Webinars, and Contact Sales. The main content area features links for MySQL Enterprise Edition and MySQL NDB Cluster CGE. A red box highlights the 'MySQL Community (GPL) Downloads' link. On the right, there is a promotional box for MySQL Enterprise Edition for Developers.

**MySQL Newsletter**  
[Subscribe »](#)  
[Archive »](#)

**Free Webinars**  
Increase your Financial Services Security with MySQL Enterprise Edition  
Thursday, May 01, 2025  
Unlocking the Power of JavaScript in MySQL: Creating Stored Programs with Ease  
Tuesday, May 13, 2025  
Getting Started with MySQL: A Beginner's Guide  
Thursday, May 15, 2025  
[More »](#)

**Contact Sales**  
USA: +1-866-221-0634  
Canada: +1-866-221-0634  
  
Germany: +49 89 143 01280  
France: +33 1 57 60 83 57  
Italy: +39 02 249 59 120  
UK: +44 207 553 8447  
  
Japan: 0120-065556  
China: 400 669 0561  
India: 0008001005870  
  
[More Countries »](#)  
[Contact Us Online »](#)


**MySQL Enterprise Edition**  
MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools and technical support for MySQL.  
[Learn More »](#)  
[Customer Download from My Oracle Support \(MOS\) »](#)  
[Trial Download from Oracle edelivery »](#)  
[Developer Download from Oracle OTN »](#)

**MySQL NDB Cluster CGE**  
MySQL NDB Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.  

- MySQL NDB Cluster
- MySQL NDB Cluster Manager
- Plus, everything in MySQL Enterprise Edition

[Learn More »](#)  
[Customer Download from My Oracle Support \(MOS\) »](#)  
[Trial Download from Oracle edelivery »](#)

**MySQL Community (GPL) Downloads »**

**MySQL Enterprise Edition for Developers**  
Free for learning, developing, and prototyping.  
  
[Download Now »](#)


# MySQL8.0安裝與設定(Installation & Settings)

## MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL NDB Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench
- MySQL Installer for Windows
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

### MySQL Enterprise Edition for Developers

Free for learning, developing, and prototyping.

[Download Now »](#)

ORACLE © 2025 Oracle

[Privacy](#) / [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie 偏好](#)

# MySQL8.0安裝與設定(Installation & Settings)

MySQL Community Downloads

MySQL Installer

General Availability (GA) Releases Archives

**MySQL Installer 8.0.42**

**Note:** MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:  
8.0.42

Select Operating System:  
Microsoft Windows

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.42.0.msi)	8.0.42	2.1M	Download MD5: 48c8d3217ab5921c9c20ff3c9a57798e   Signature
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.42.0.msi)	8.0.42	353.7M	Download MD5: b0406f4ea3e5942909f6b054f9575e12   Signature

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

ORACLE © 2025 Oracle

[Privacy](#) / [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie 偏好](#)

# MySQL8.0安裝與設定(Installation & Settings)

## MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »

using my Oracle Web account

Sign Up »

for an Oracle Web account

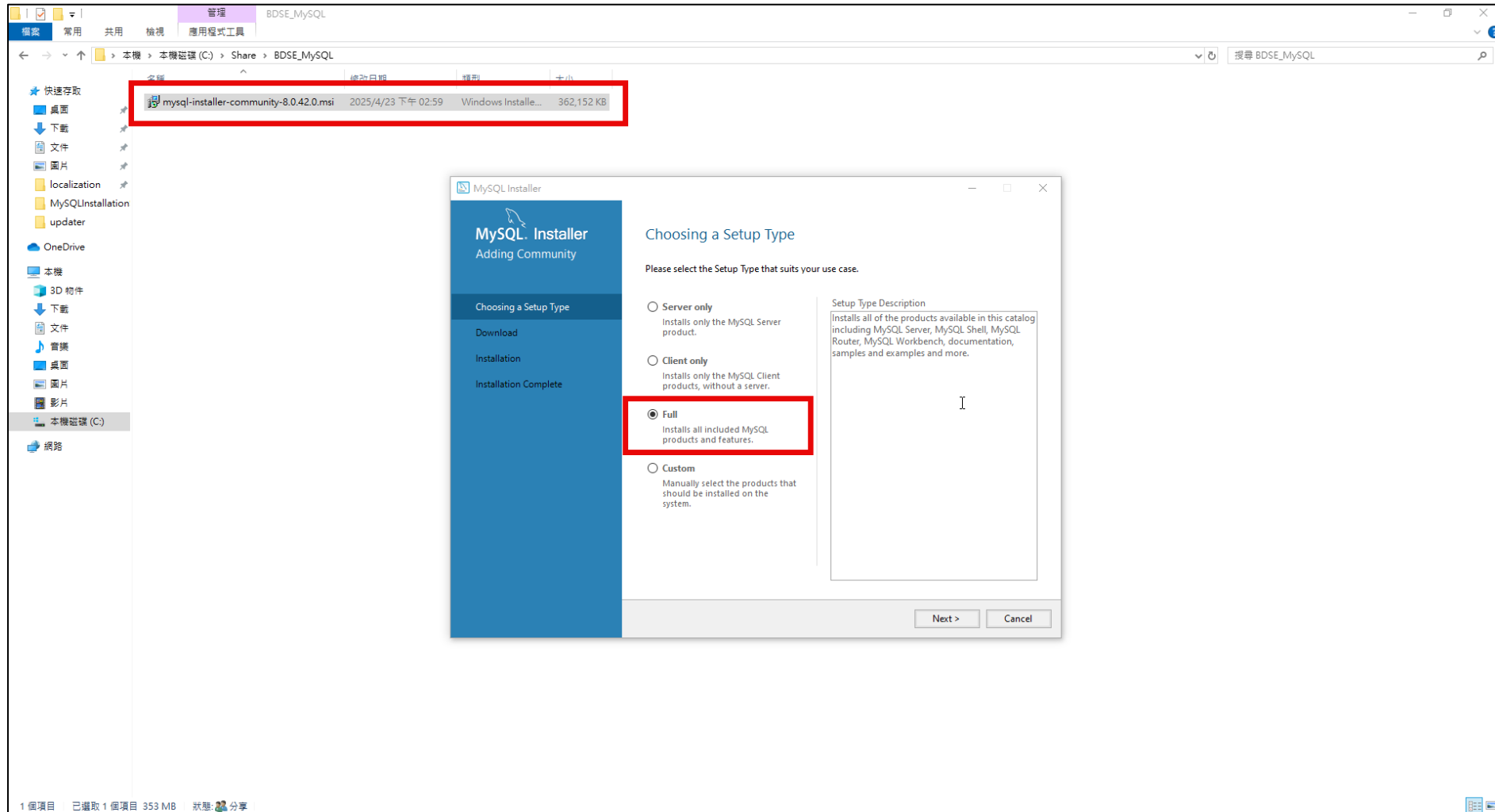
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

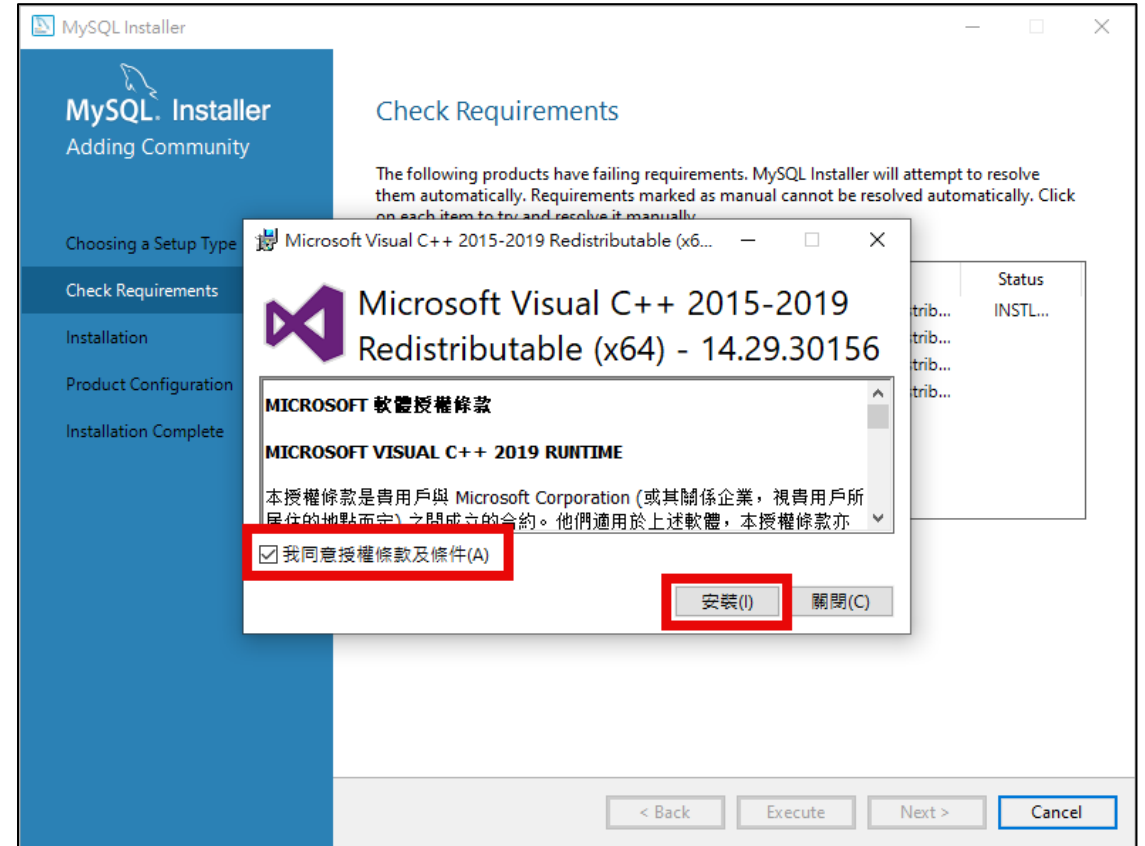
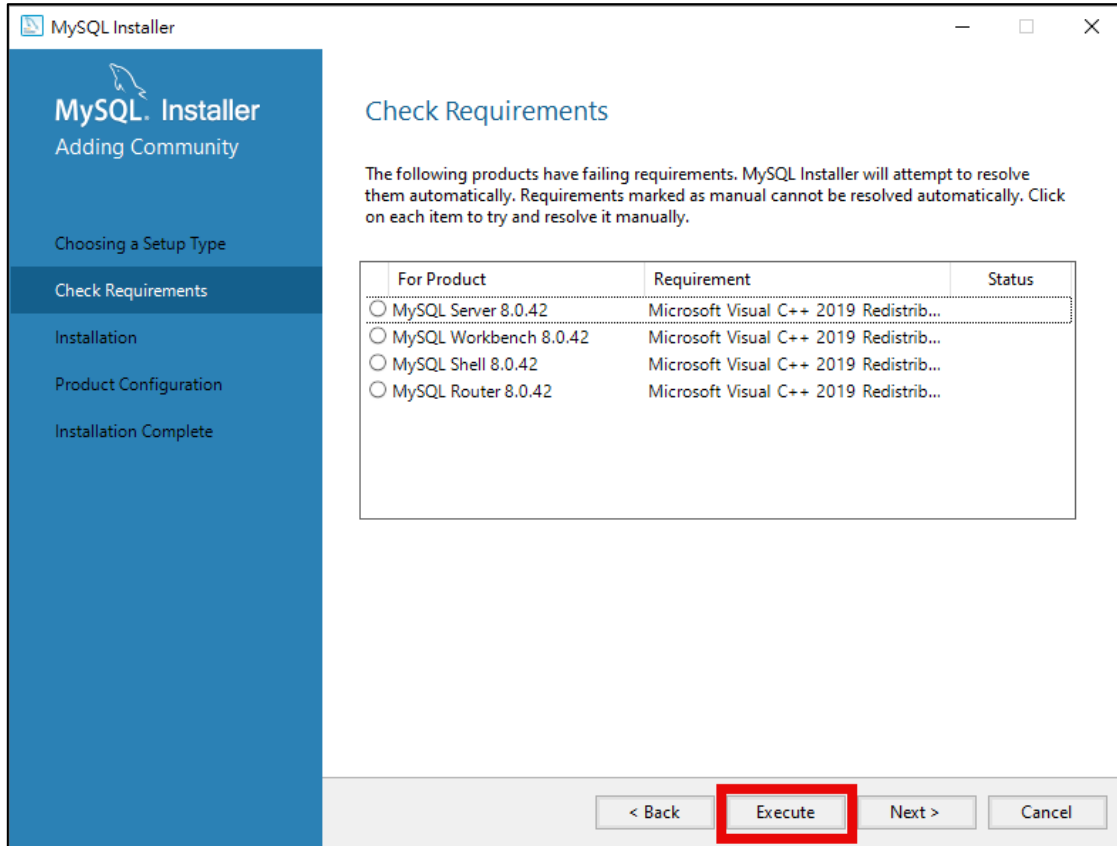
ORACLE © 2023 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie 偏好

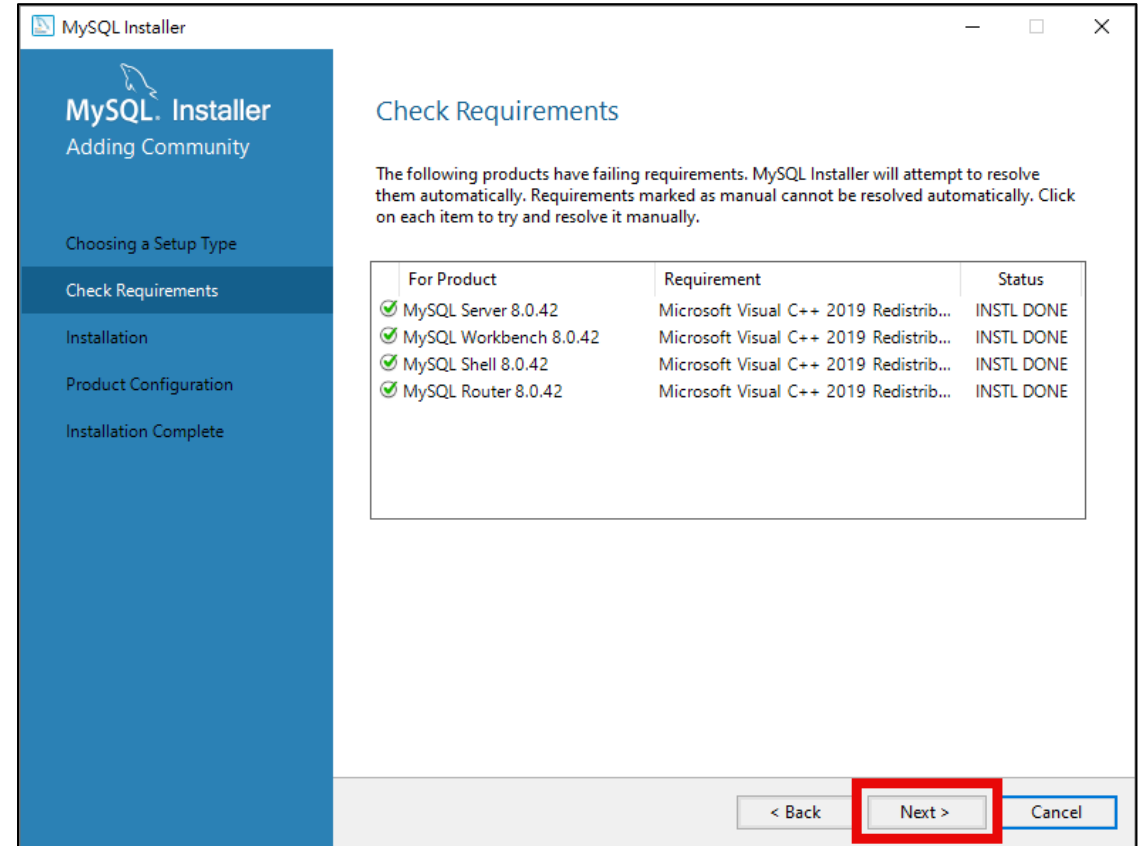
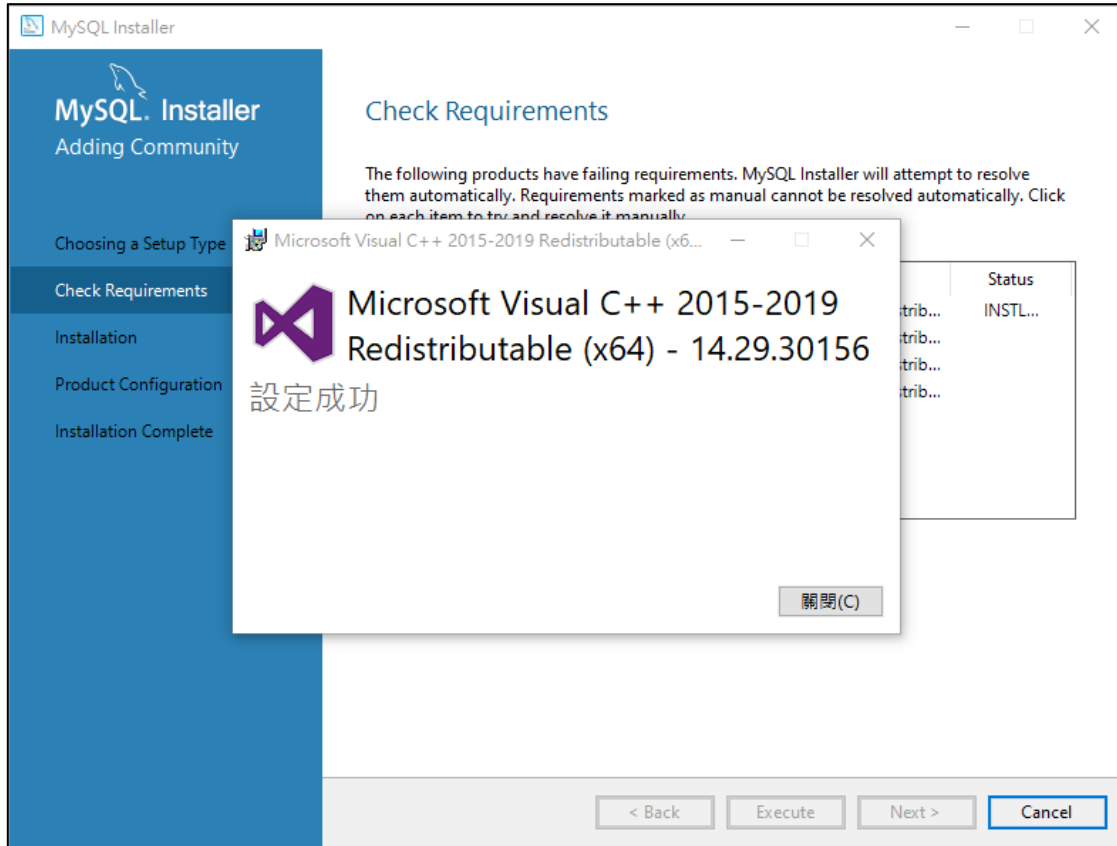
# MySQL8.0安裝與設定(Installation & Settings)



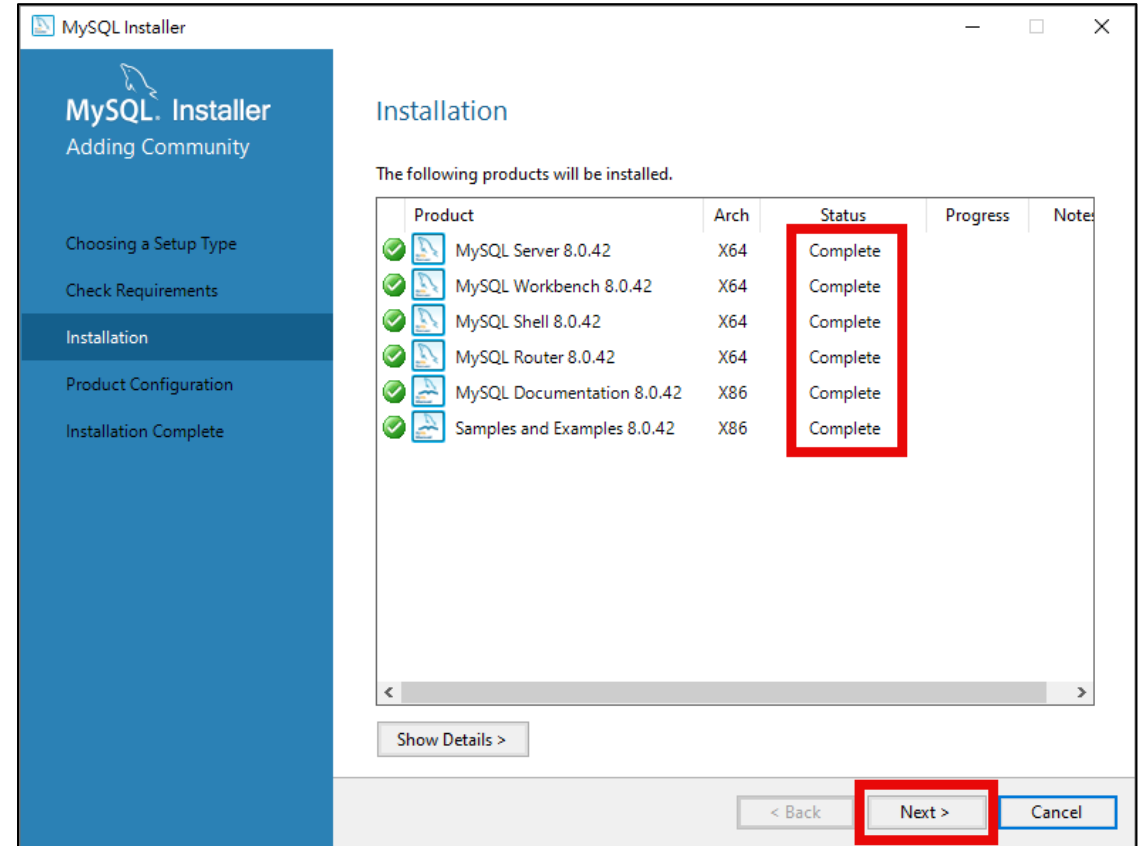
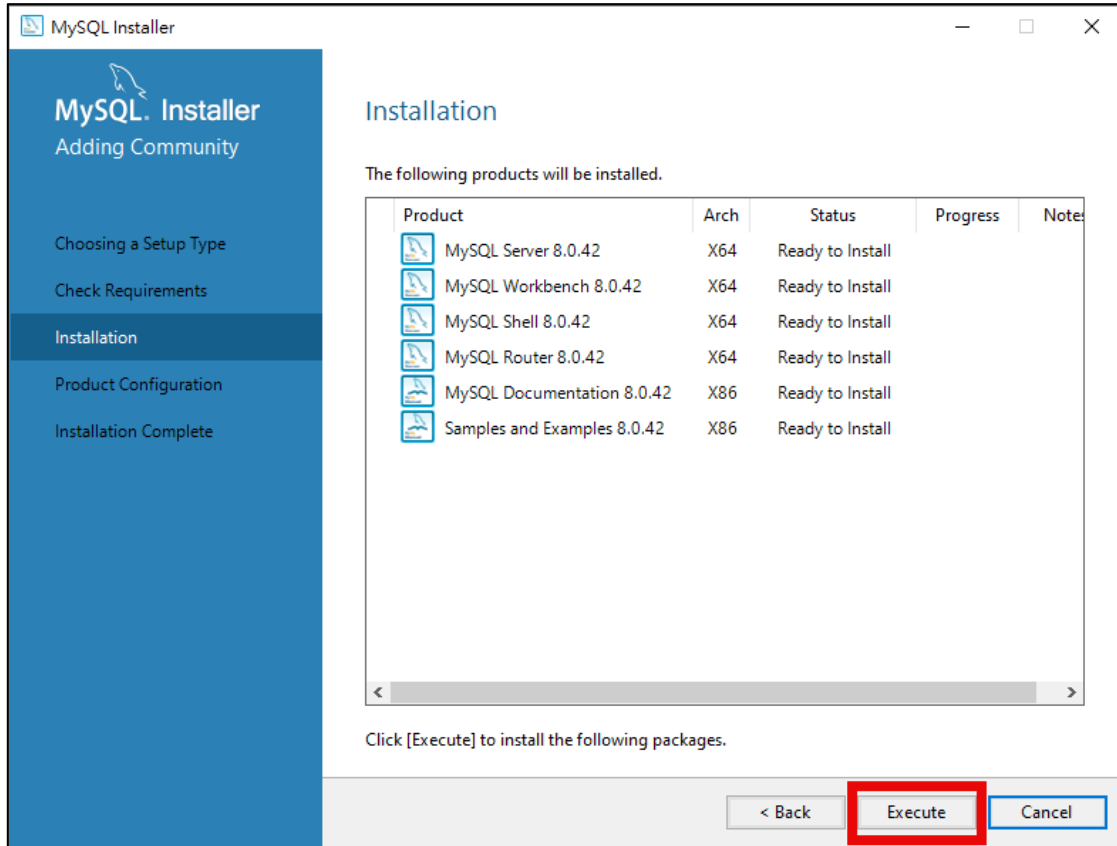
# MySQL8.0安裝與設定(Installation & Settings)



# MySQL8.0安裝與設定(Installation & Settings)

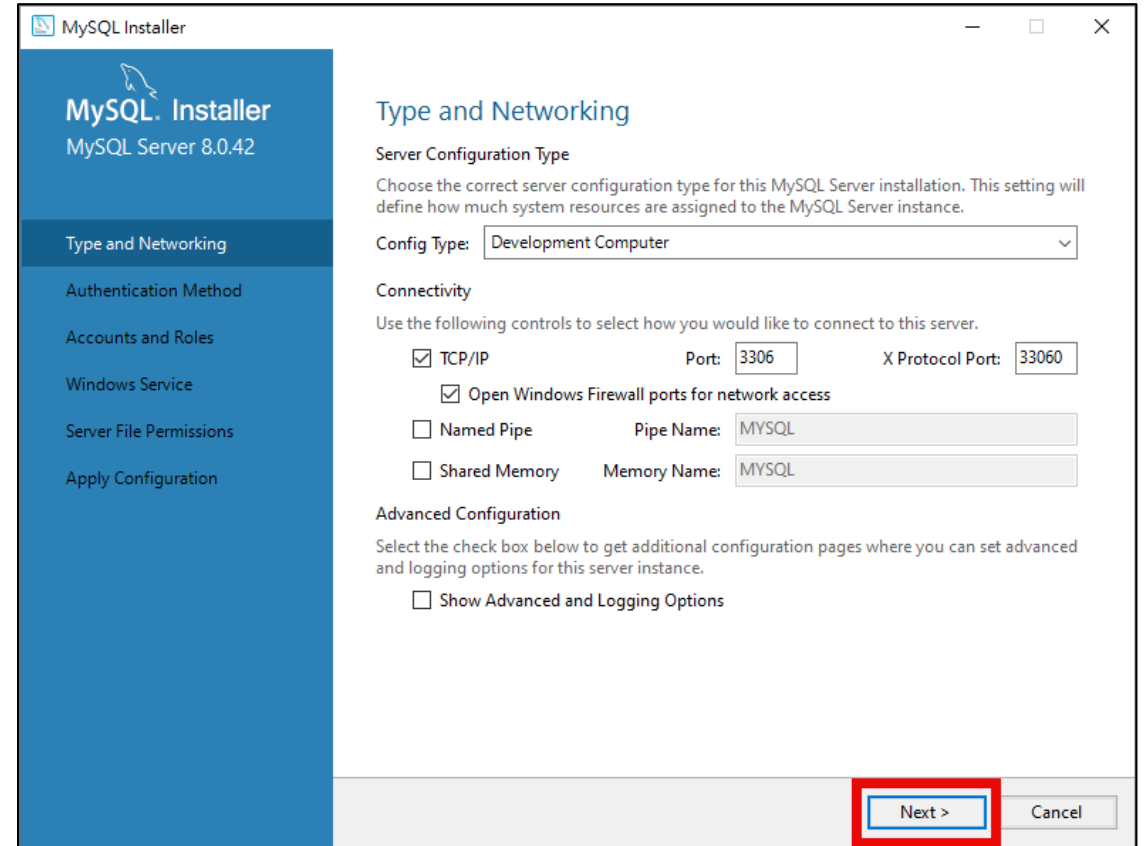
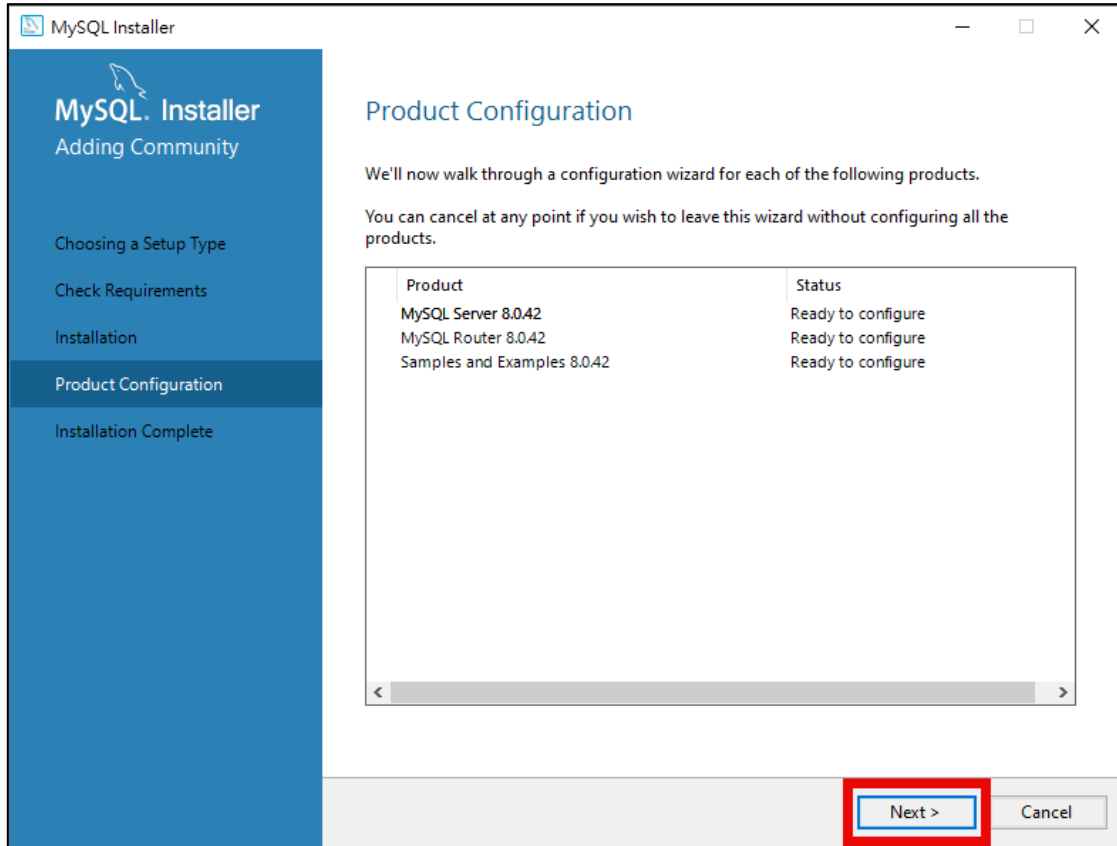


# MySQL8.0安裝與設定(Installation & Settings)



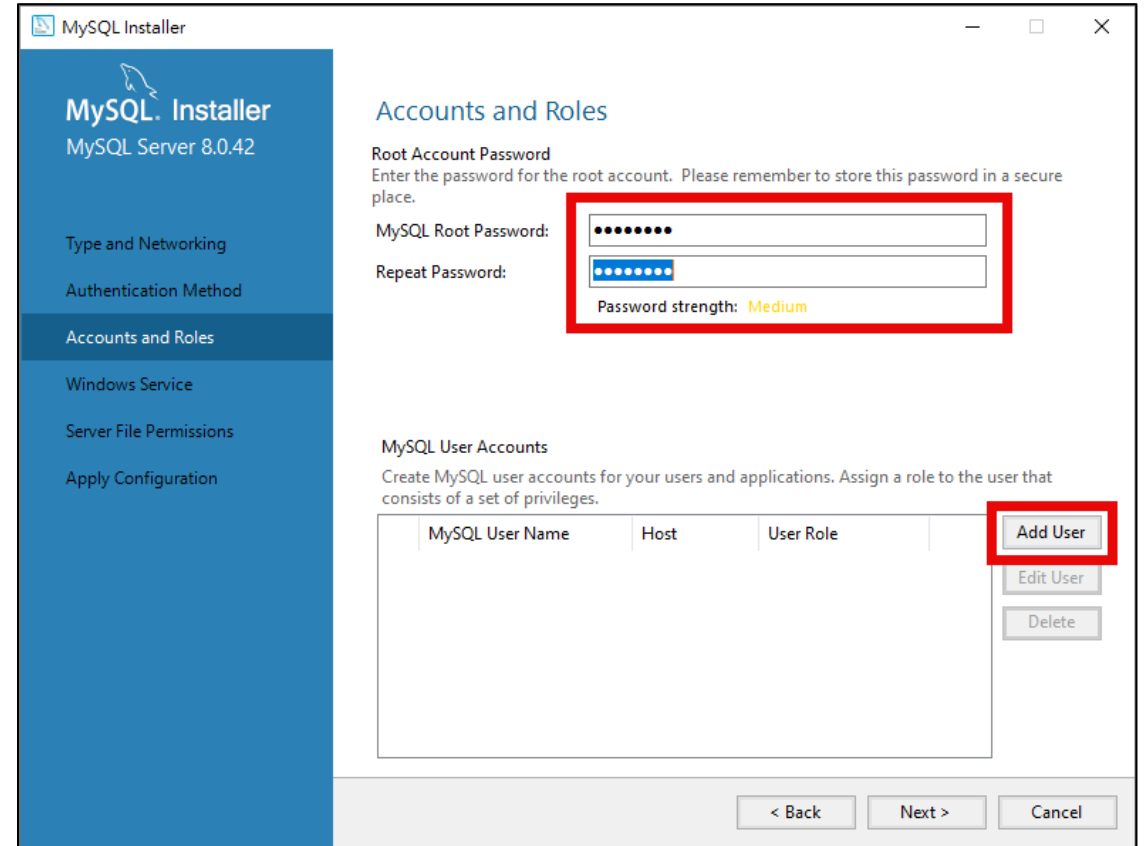
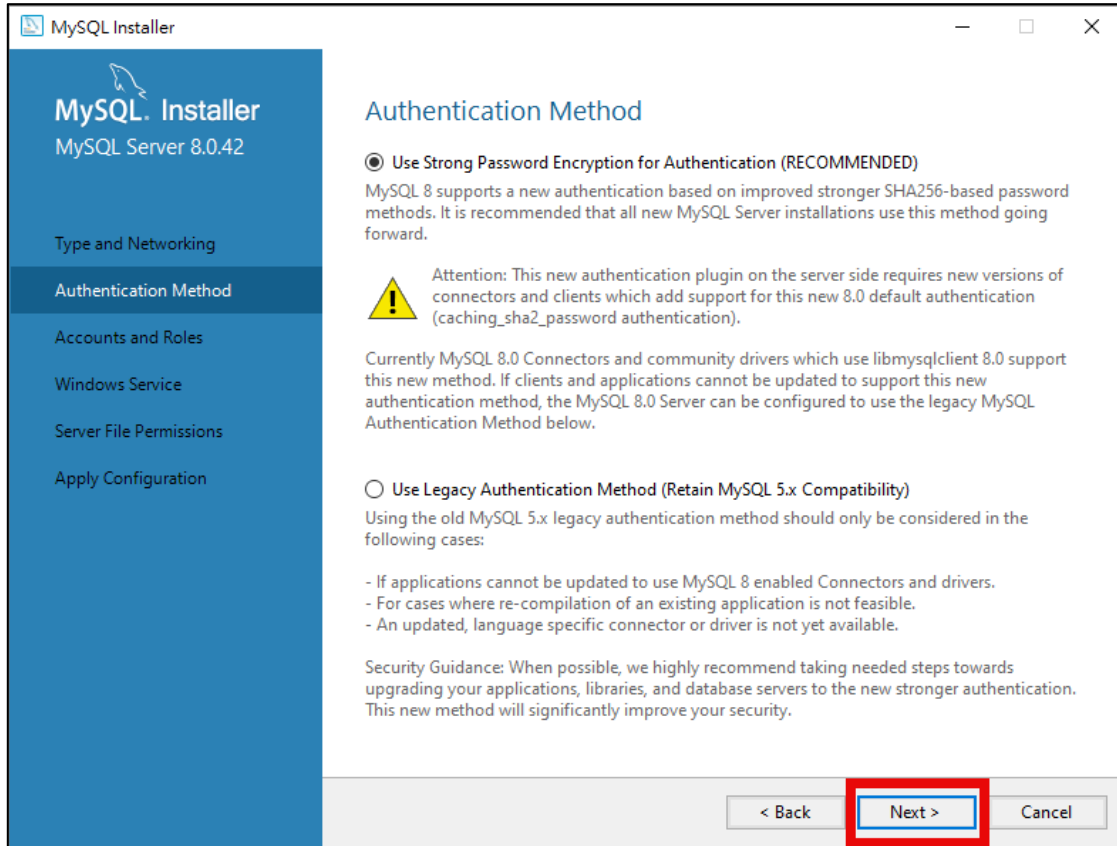
通常這裡會花一點時間...

# MySQL8.0安裝與設定(Installation & Settings)

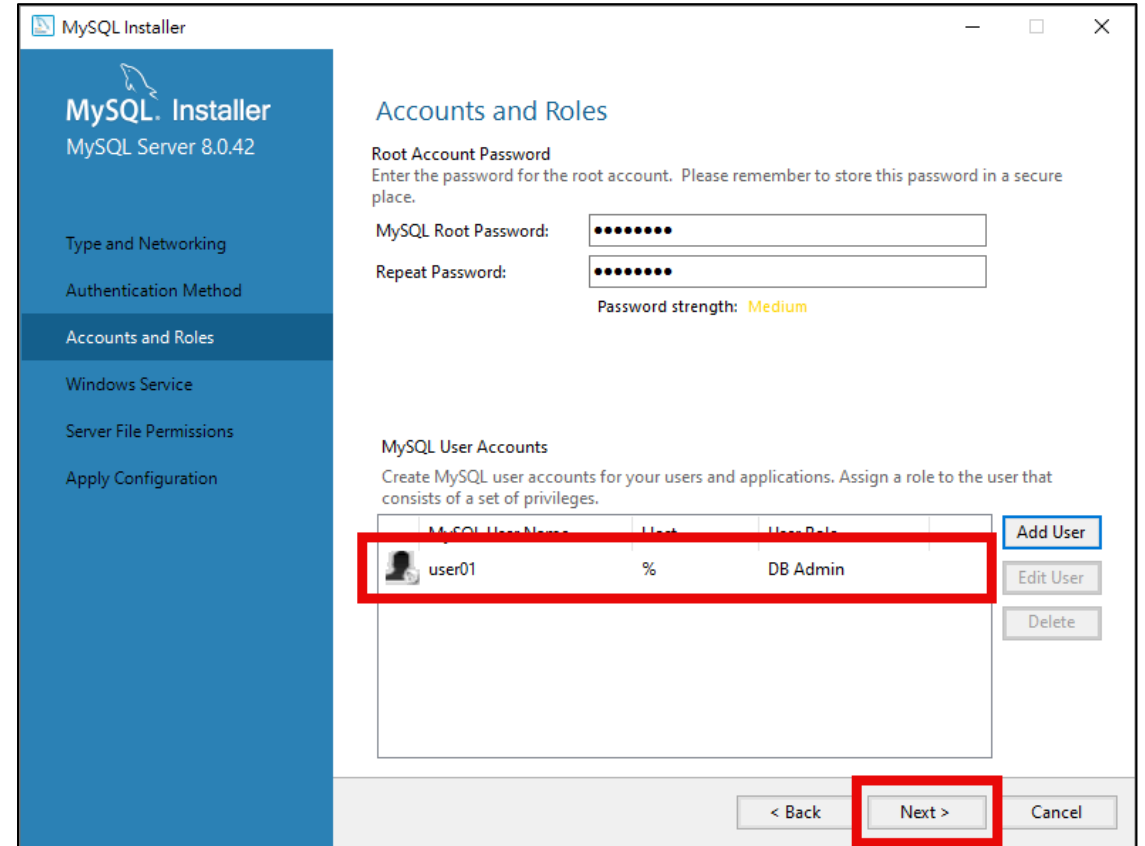
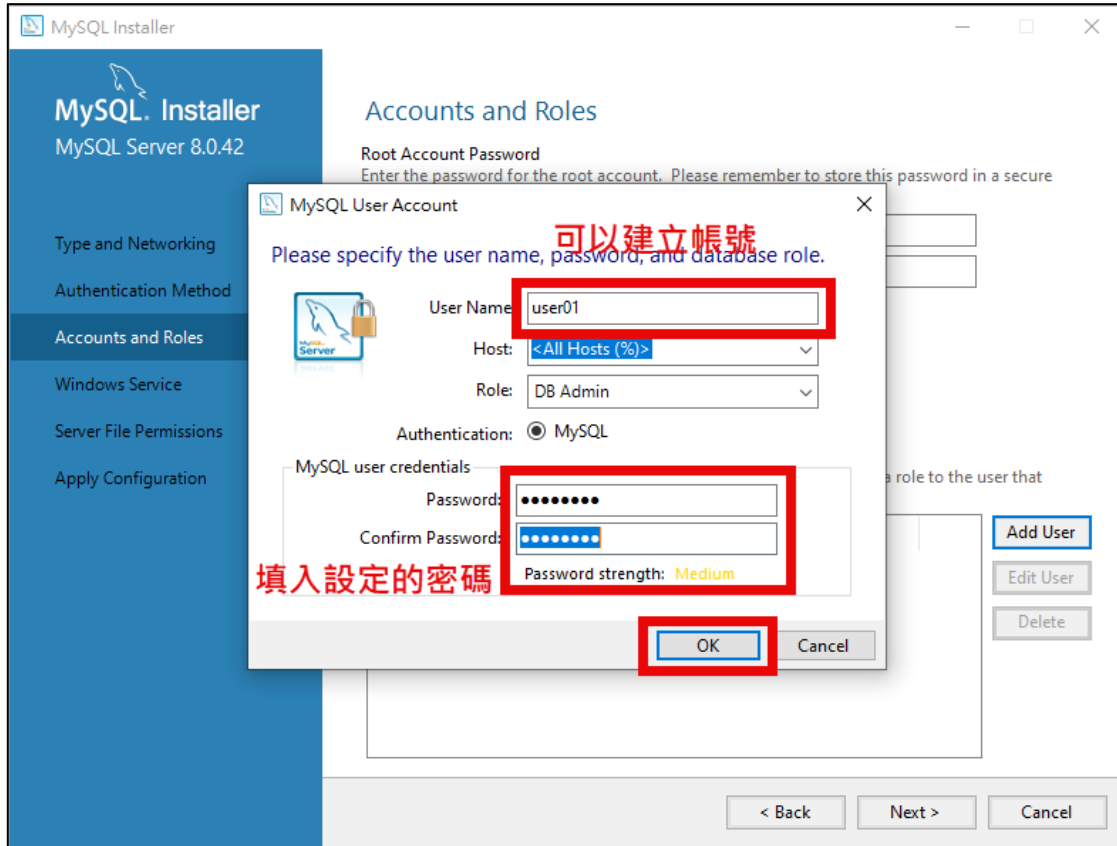


# MySQL8.0安裝與設定(Installation & Settings)

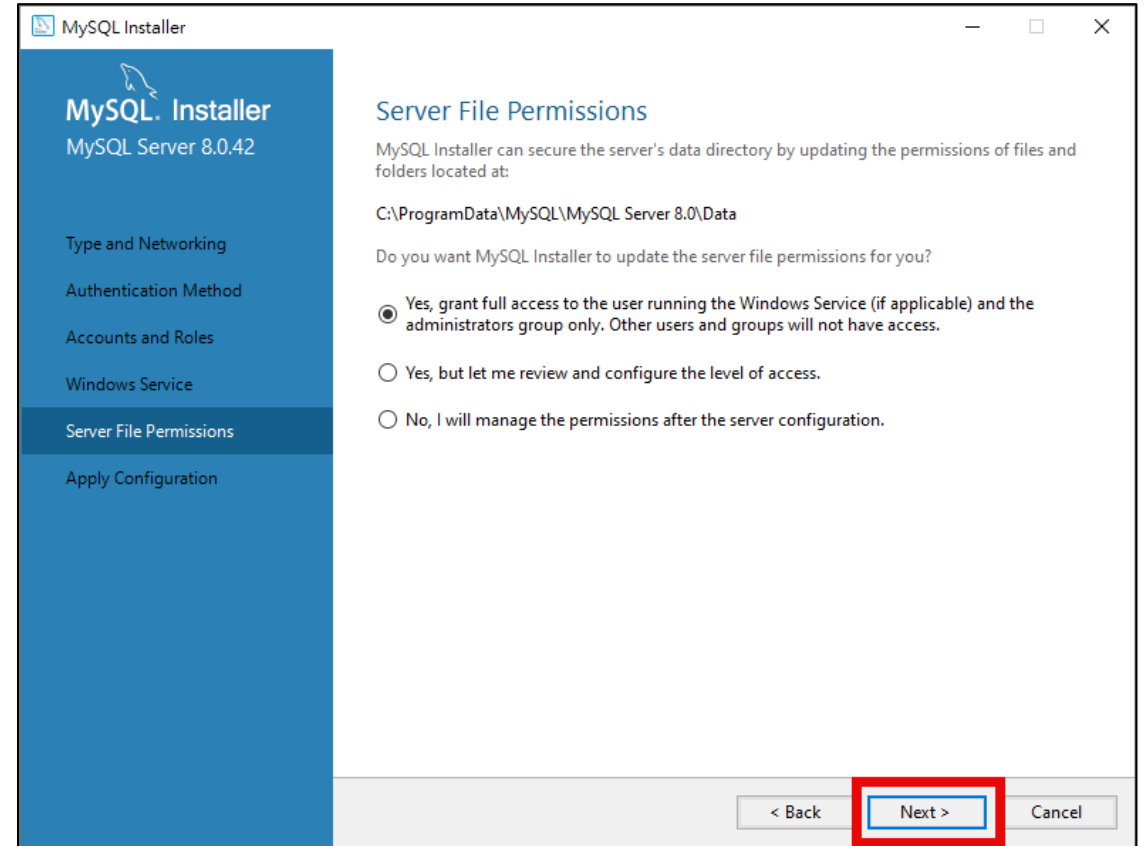
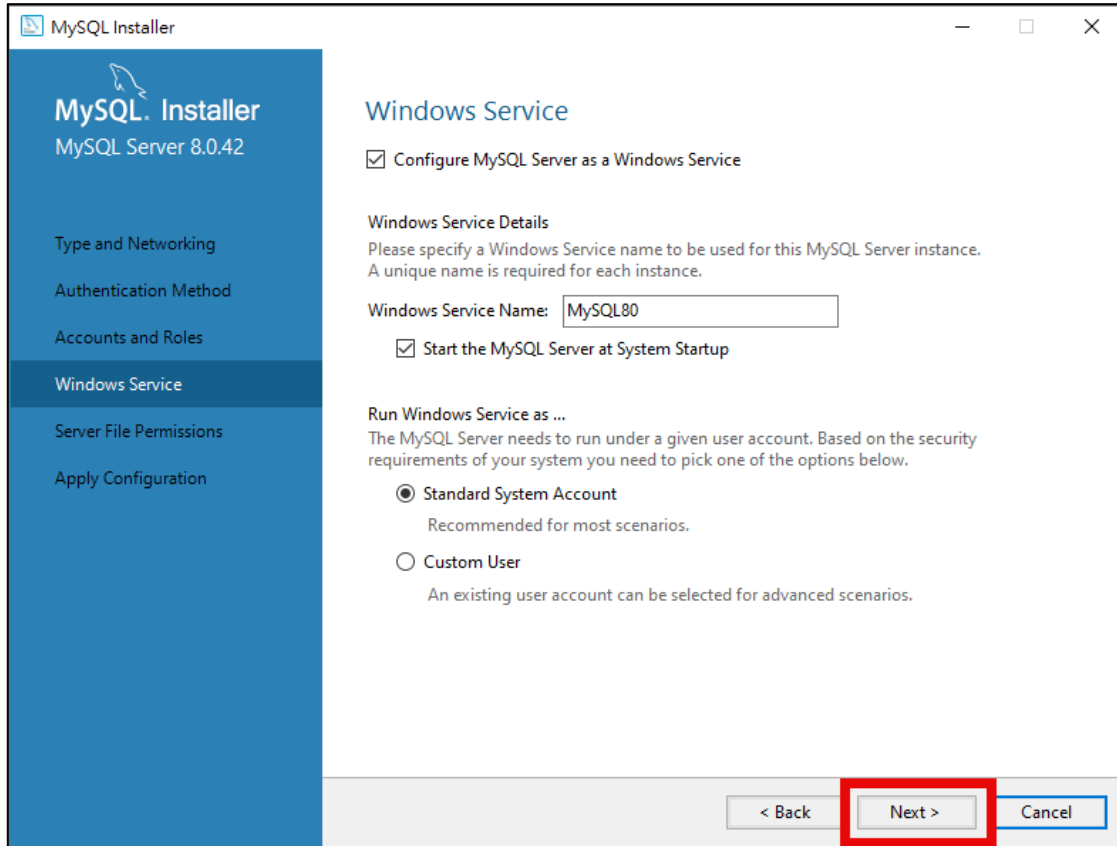
建議密碼：P@ssw0rd



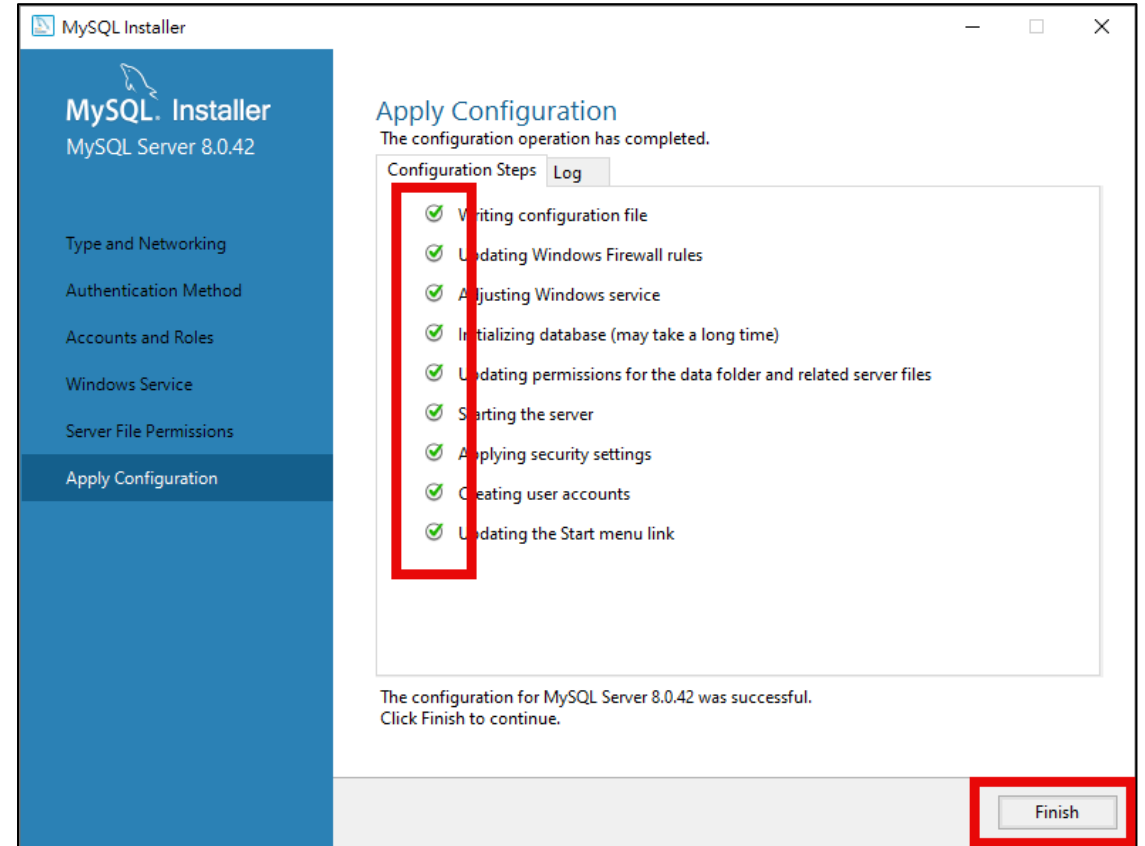
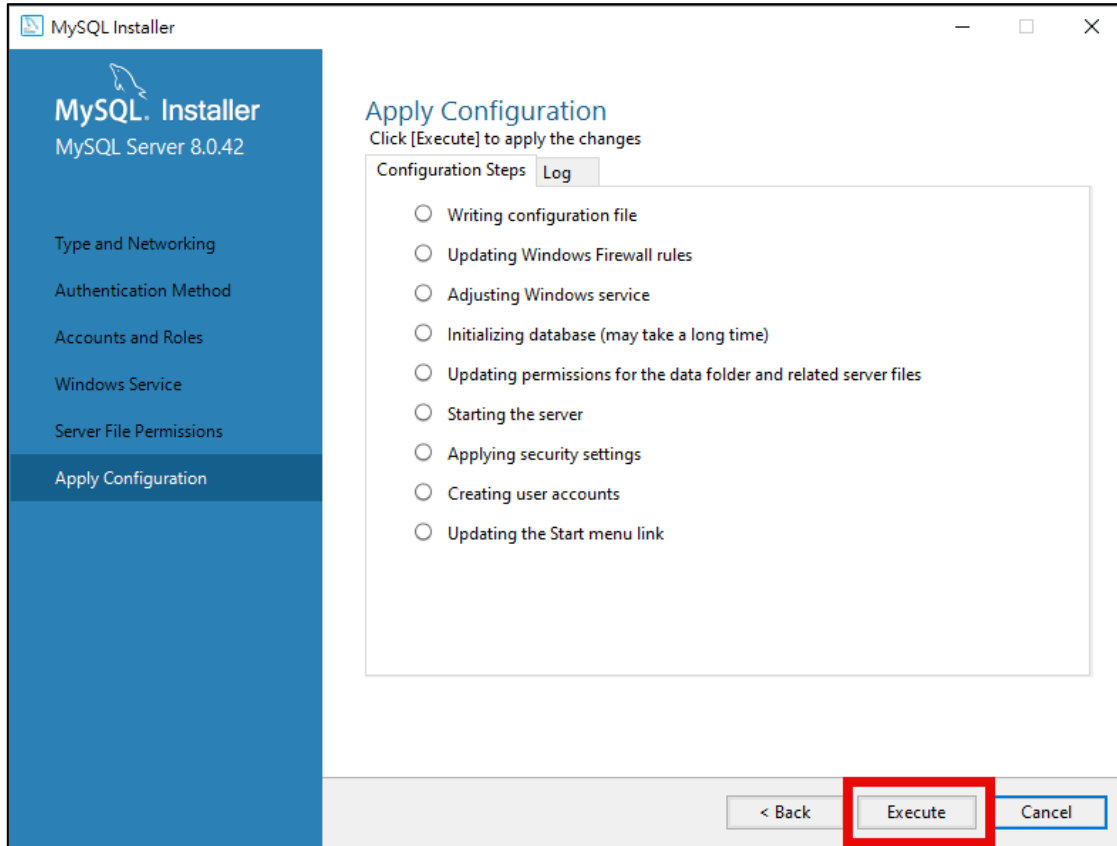
# MySQL8.0安裝與設定(Installation & Settings)



# MySQL8.0安裝與設定(Installation & Settings)

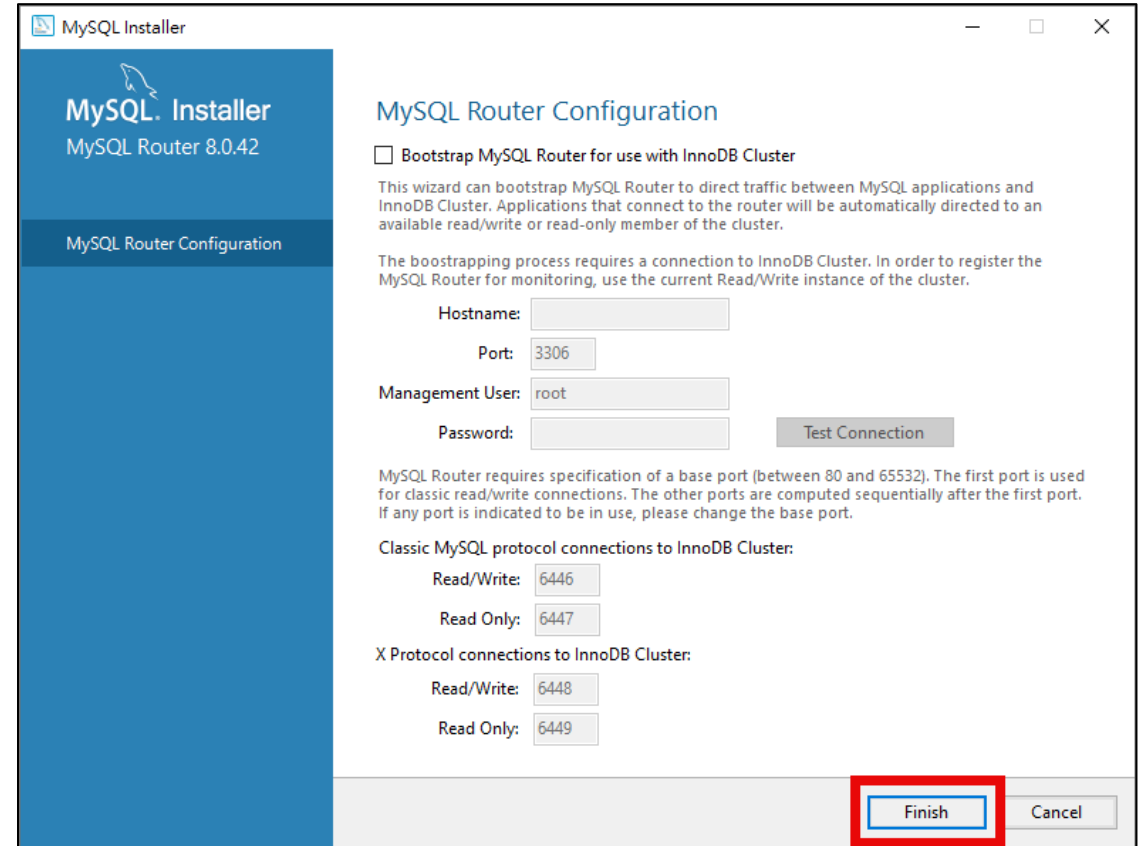
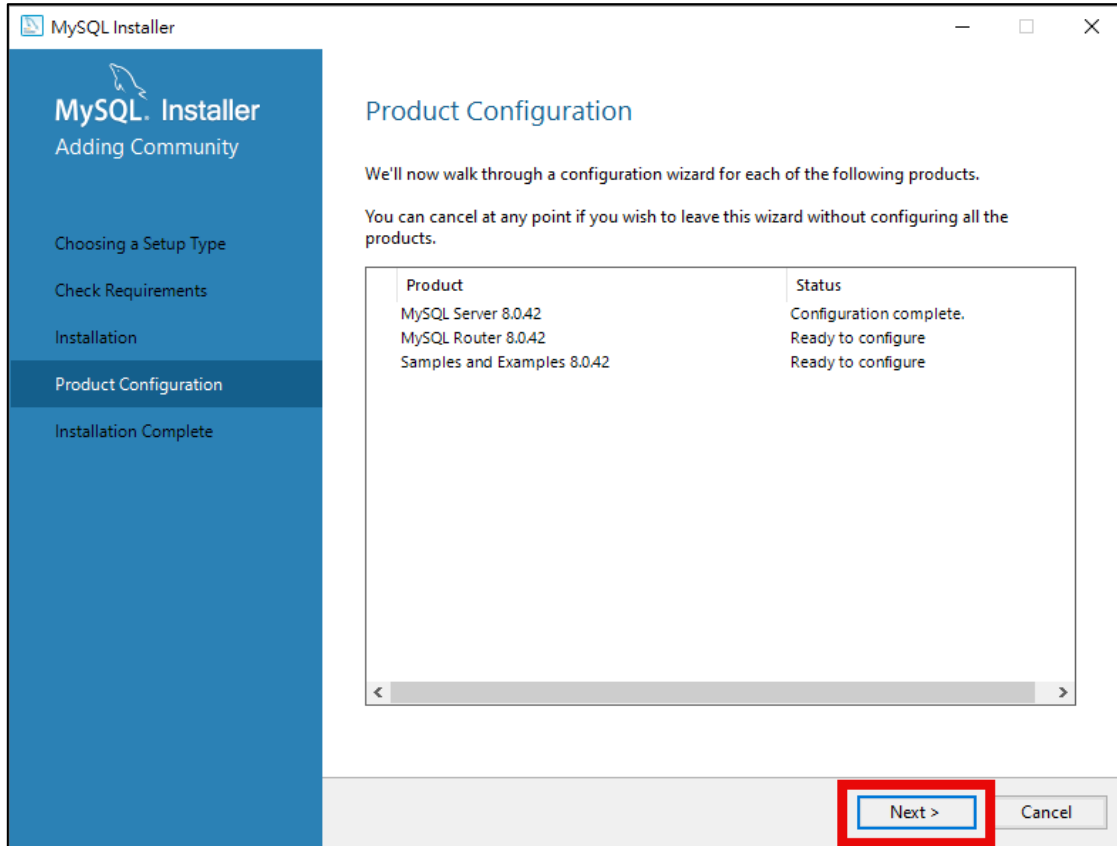


# MySQL8.0安裝與設定(Installation & Settings)

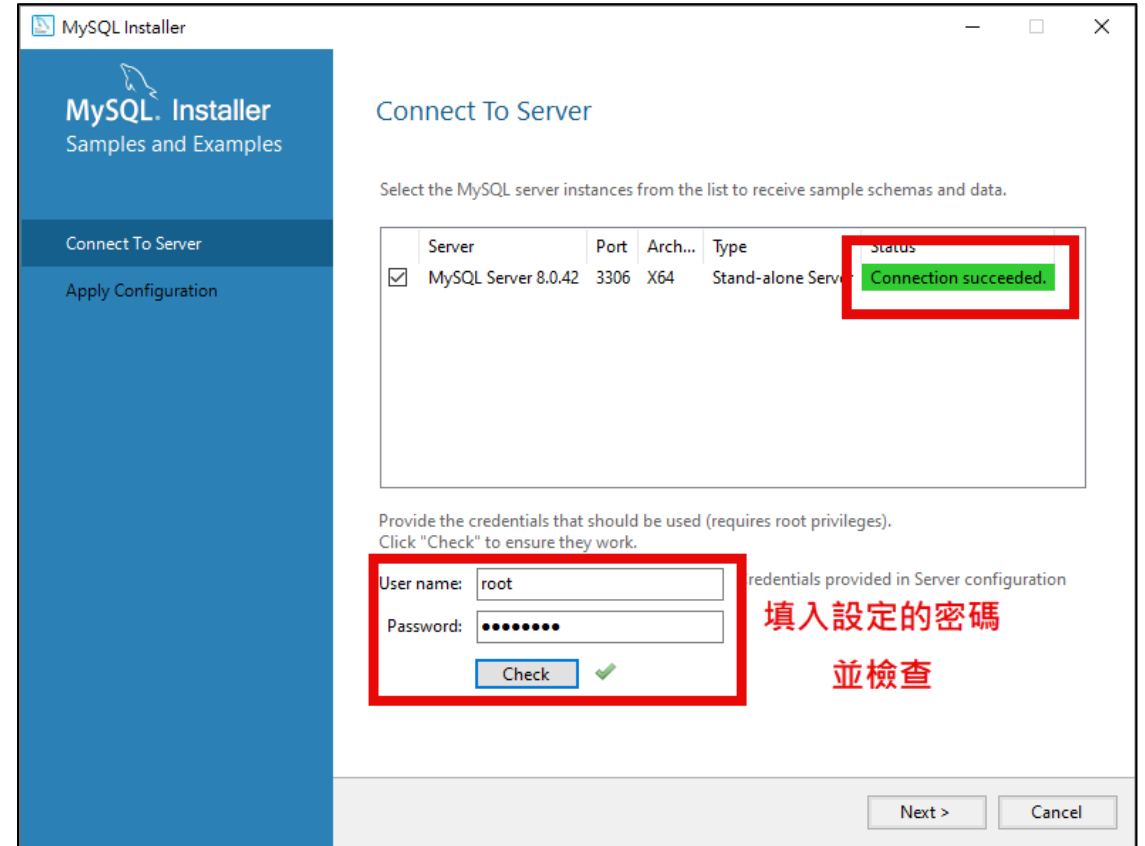
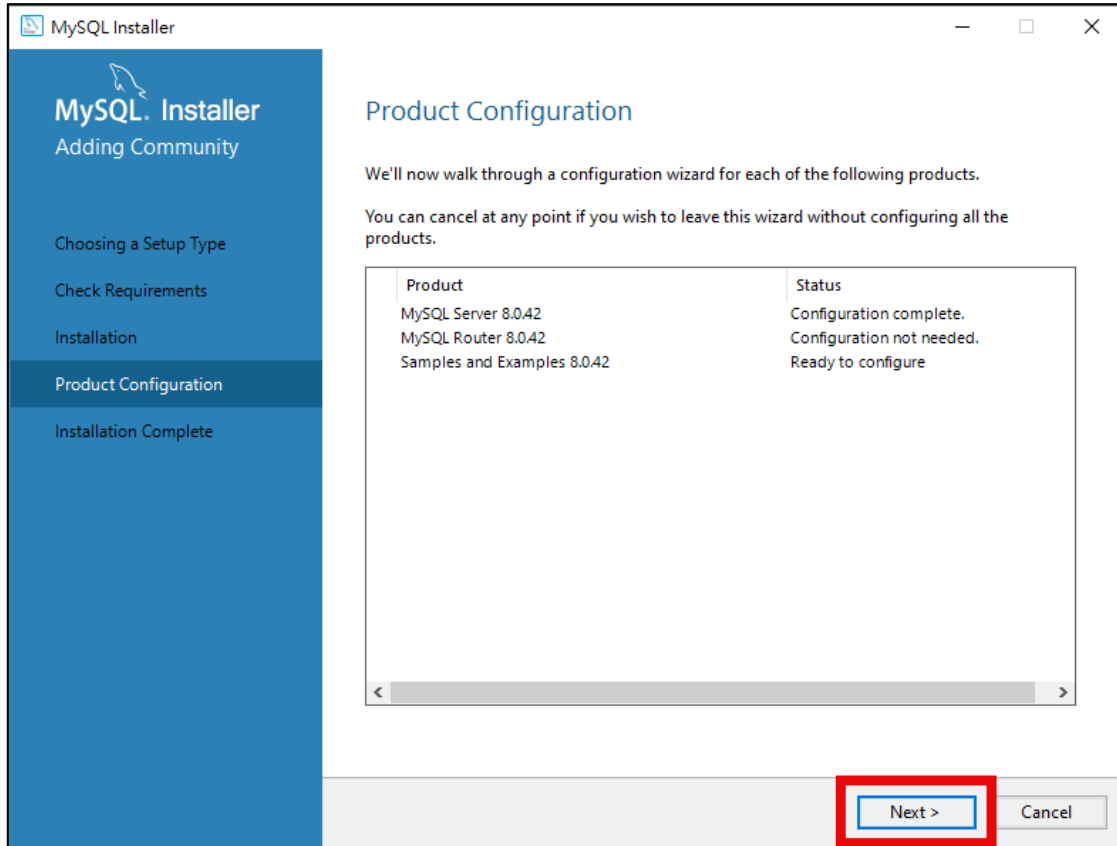


通常這裡會花一點時間...

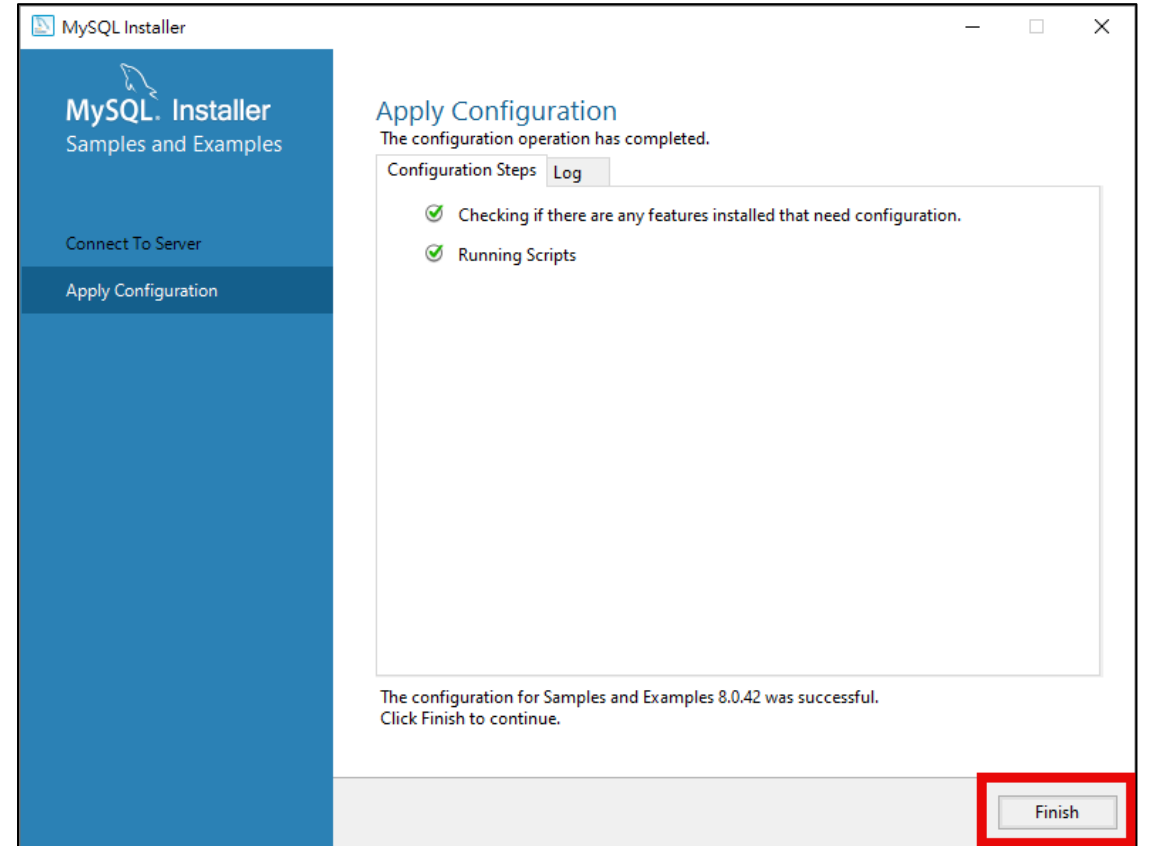
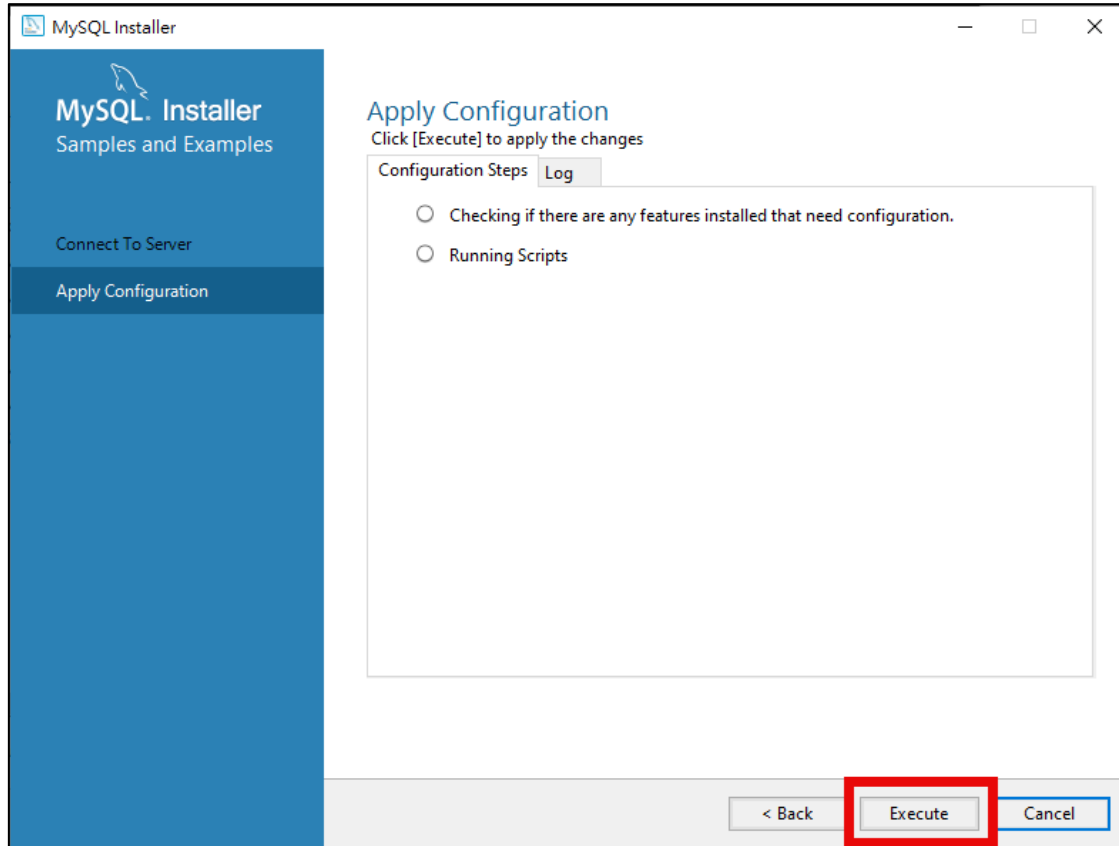
# MySQL8.0安裝與設定(Installation & Settings)



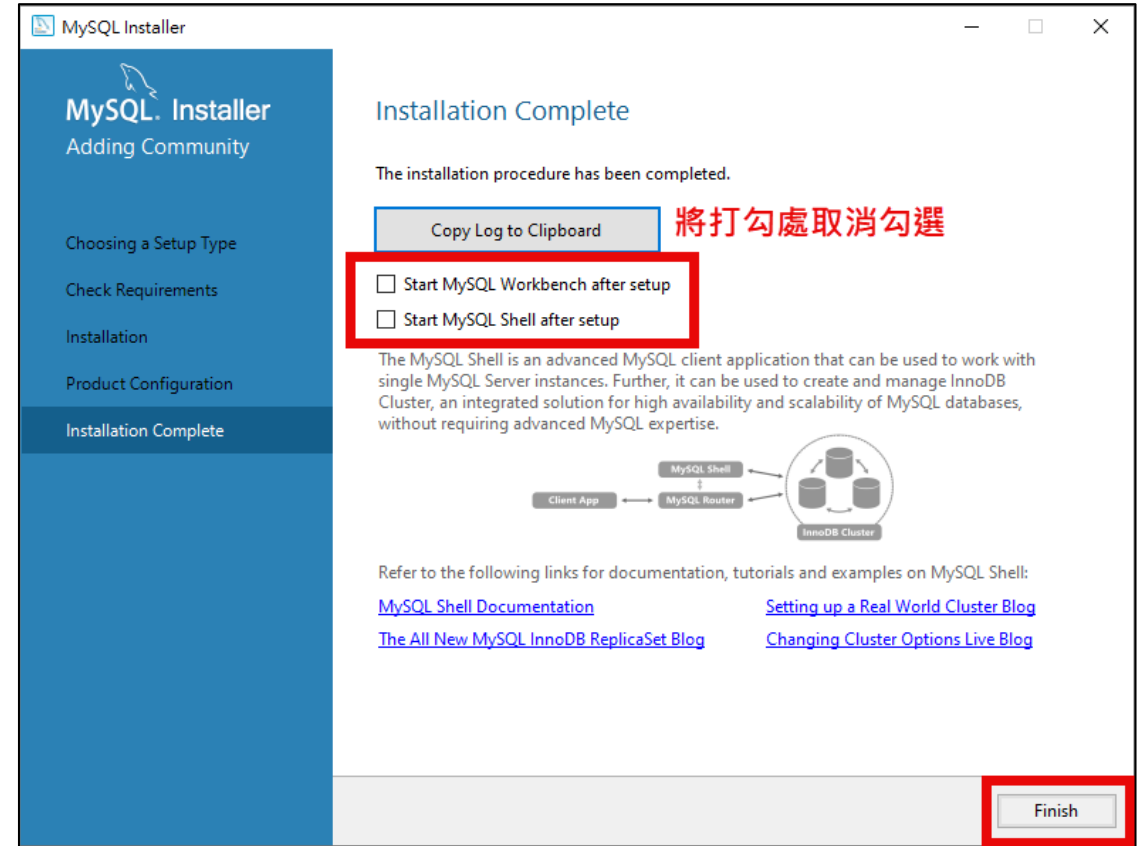
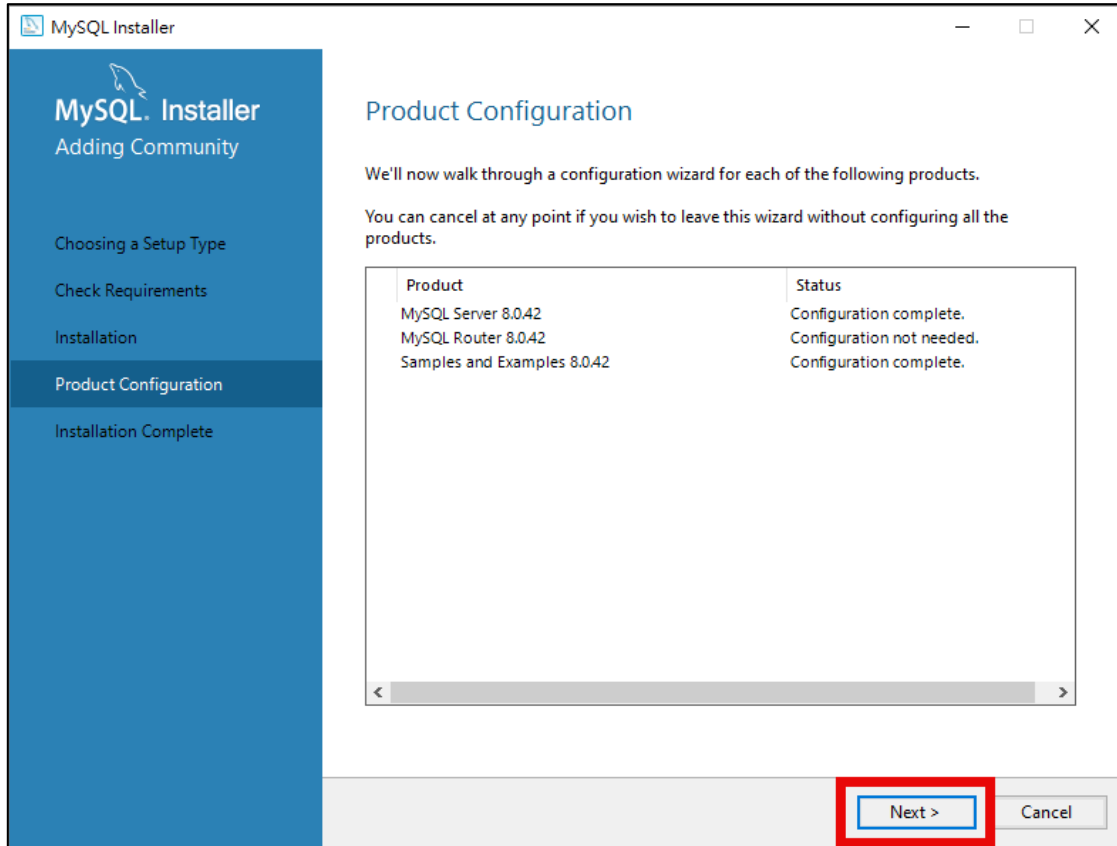
# MySQL8.0安裝與設定(Installation & Settings)



# MySQL8.0安裝與設定(Installation & Settings)

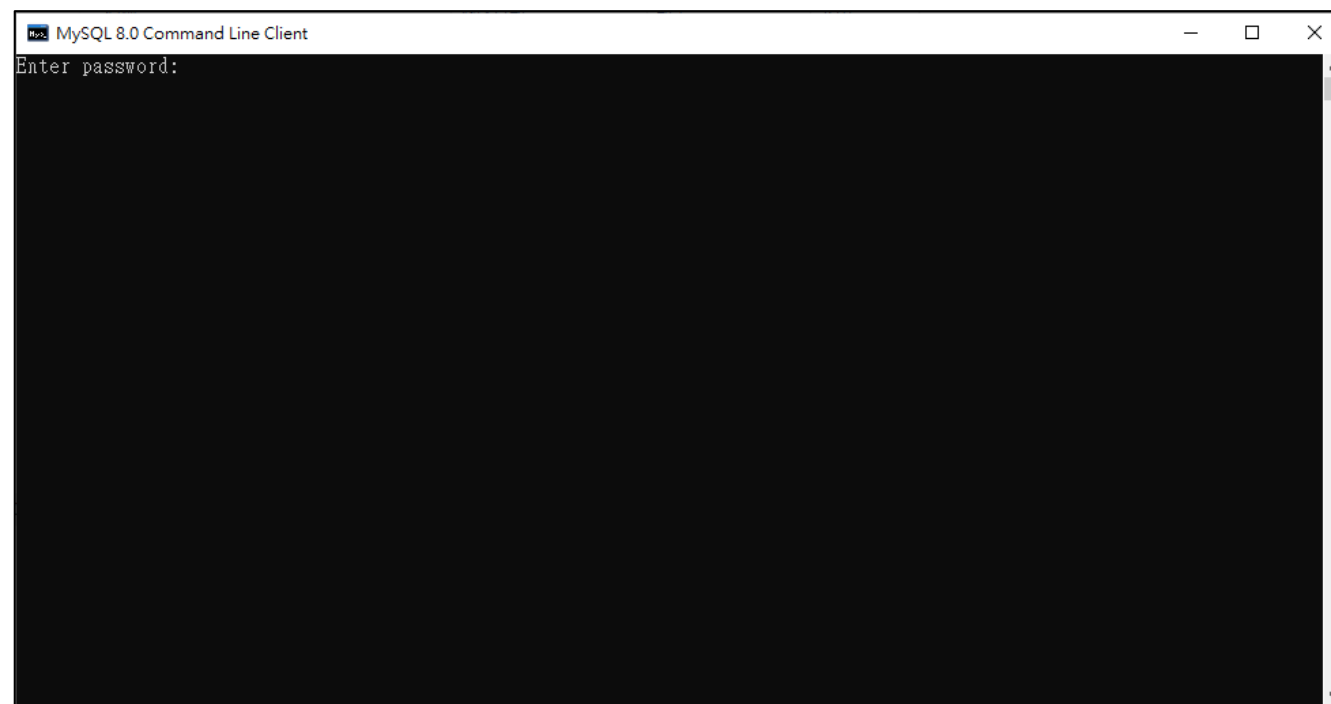
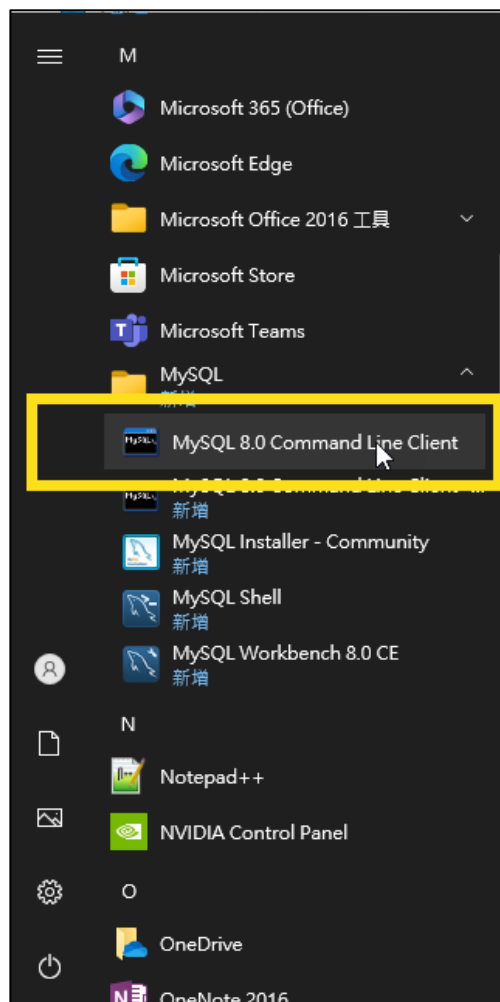


# MySQL8.0安裝與設定(Installation & Settings)



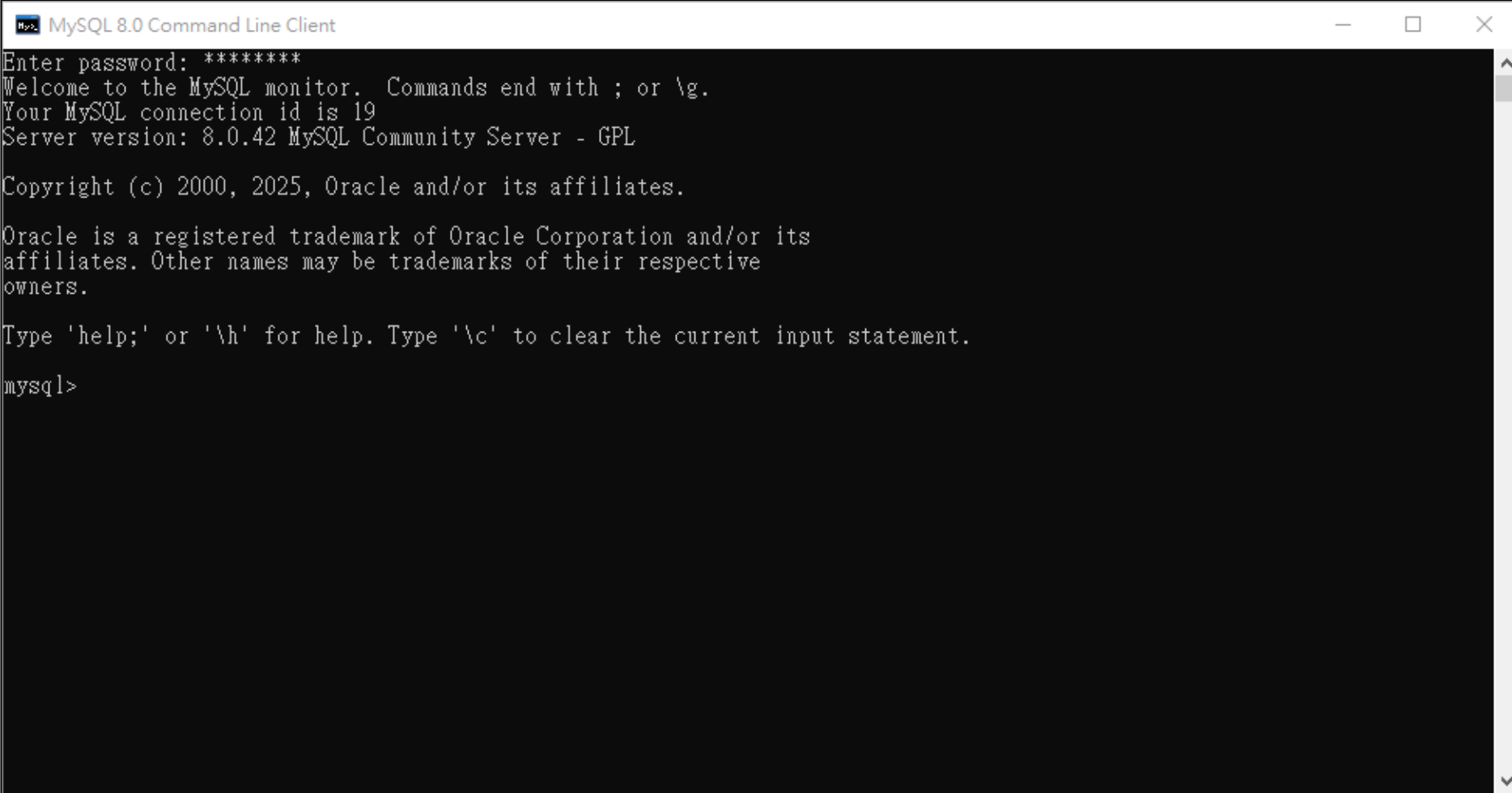
# MySQL8.0安裝與設定(Installation & Settings)

## ● 連線檢測



# MySQL8.0安裝與設定(Installation & Settings)

---



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.42 MySQL Community Server - GPL

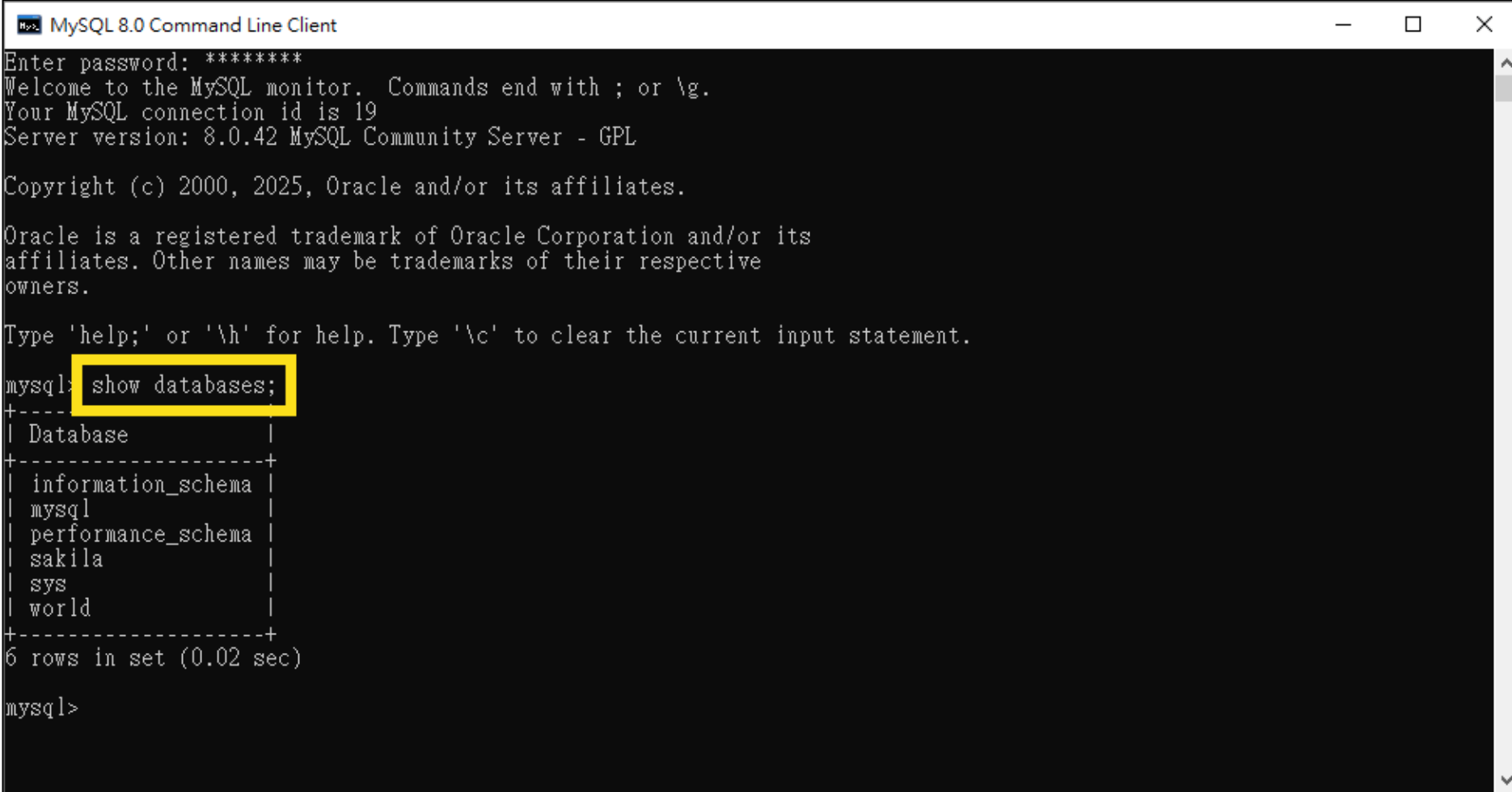
Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

# MySQL8.0安裝與設定(Installation & Settings)



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

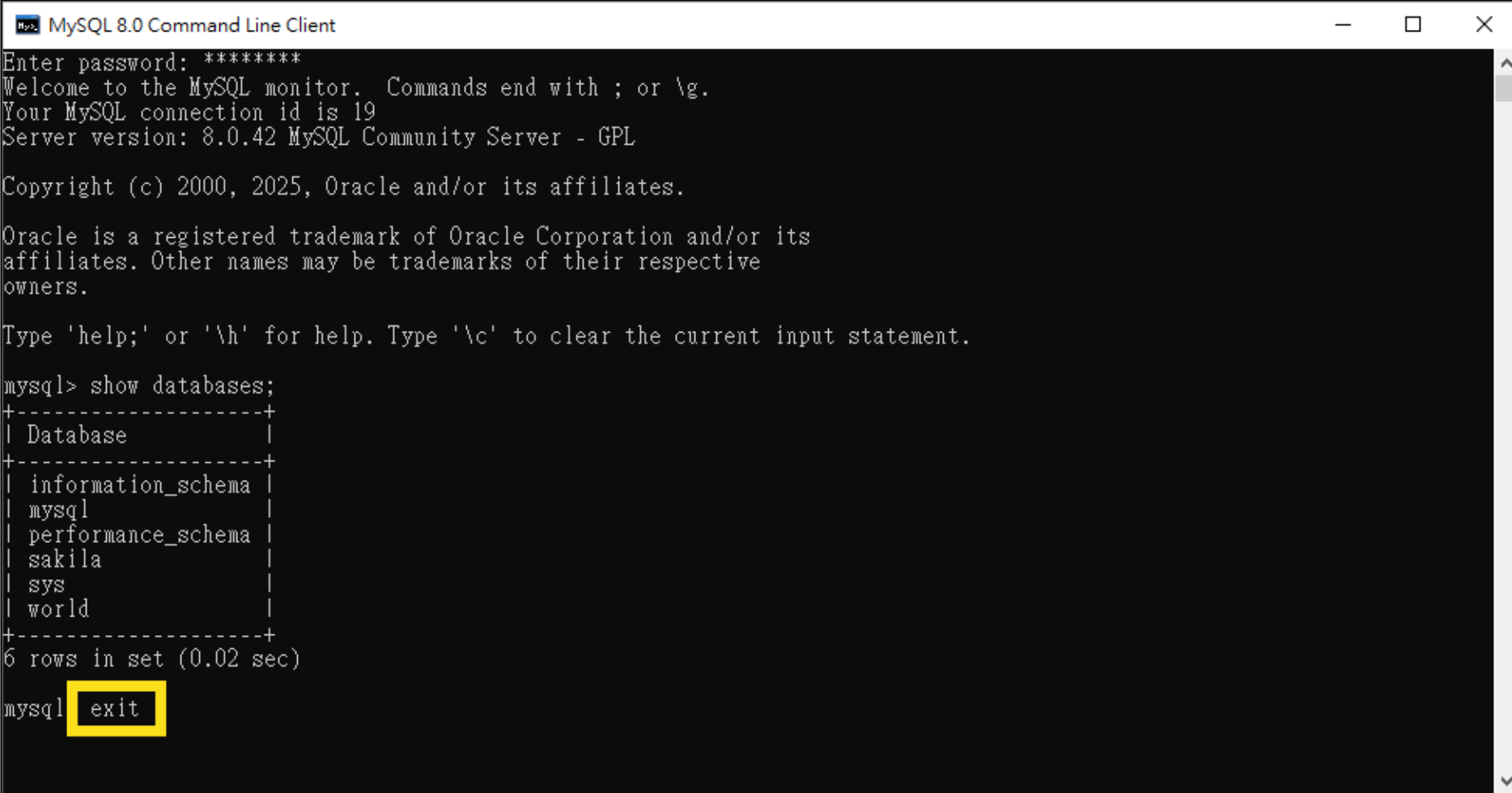
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sakila     |
| sys        |
| world      |
+-----+
6 rows in set (0.02 sec)

mysql>
```

# MySQL8.0安裝與設定(Installation & Settings)



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

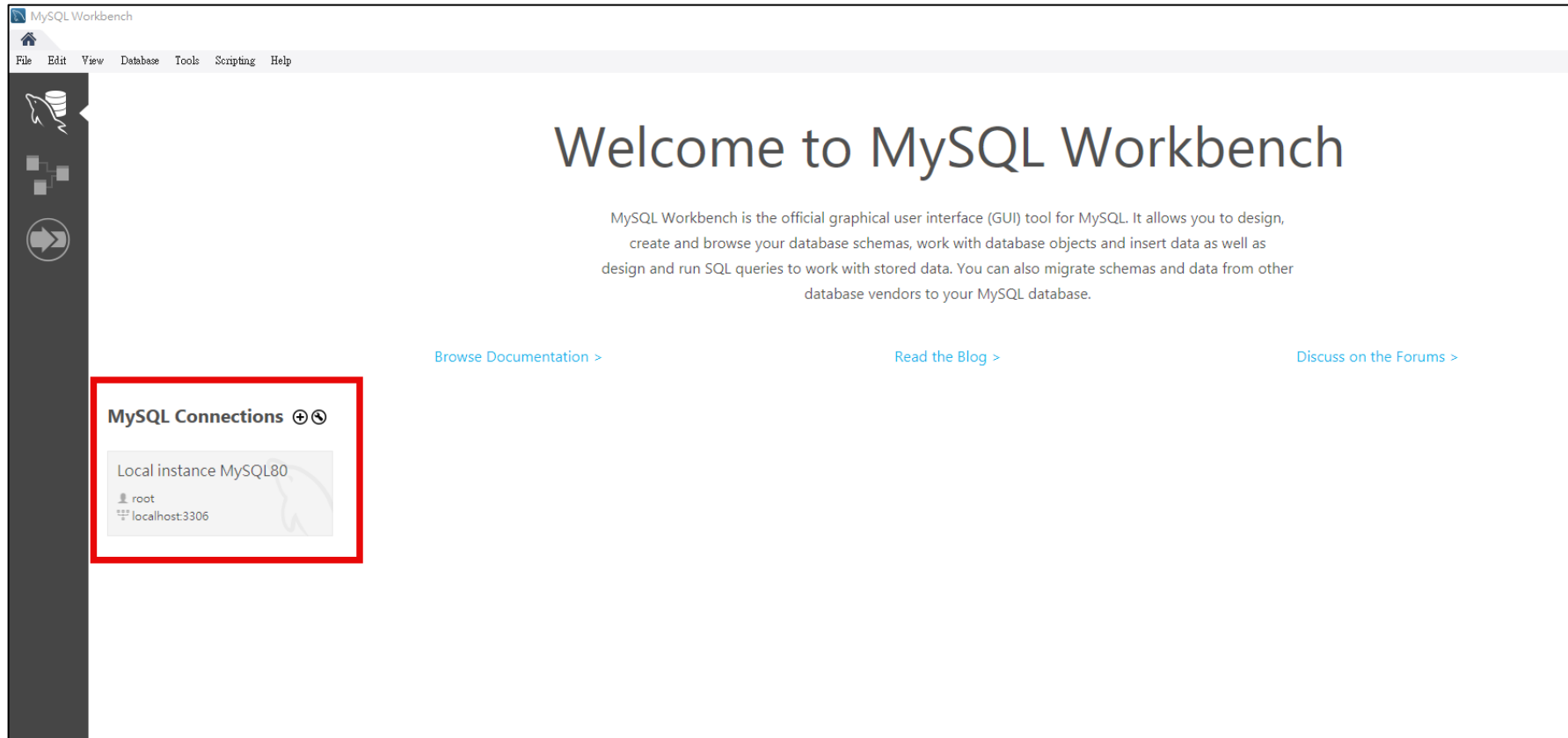
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
6 rows in set (0.02 sec)

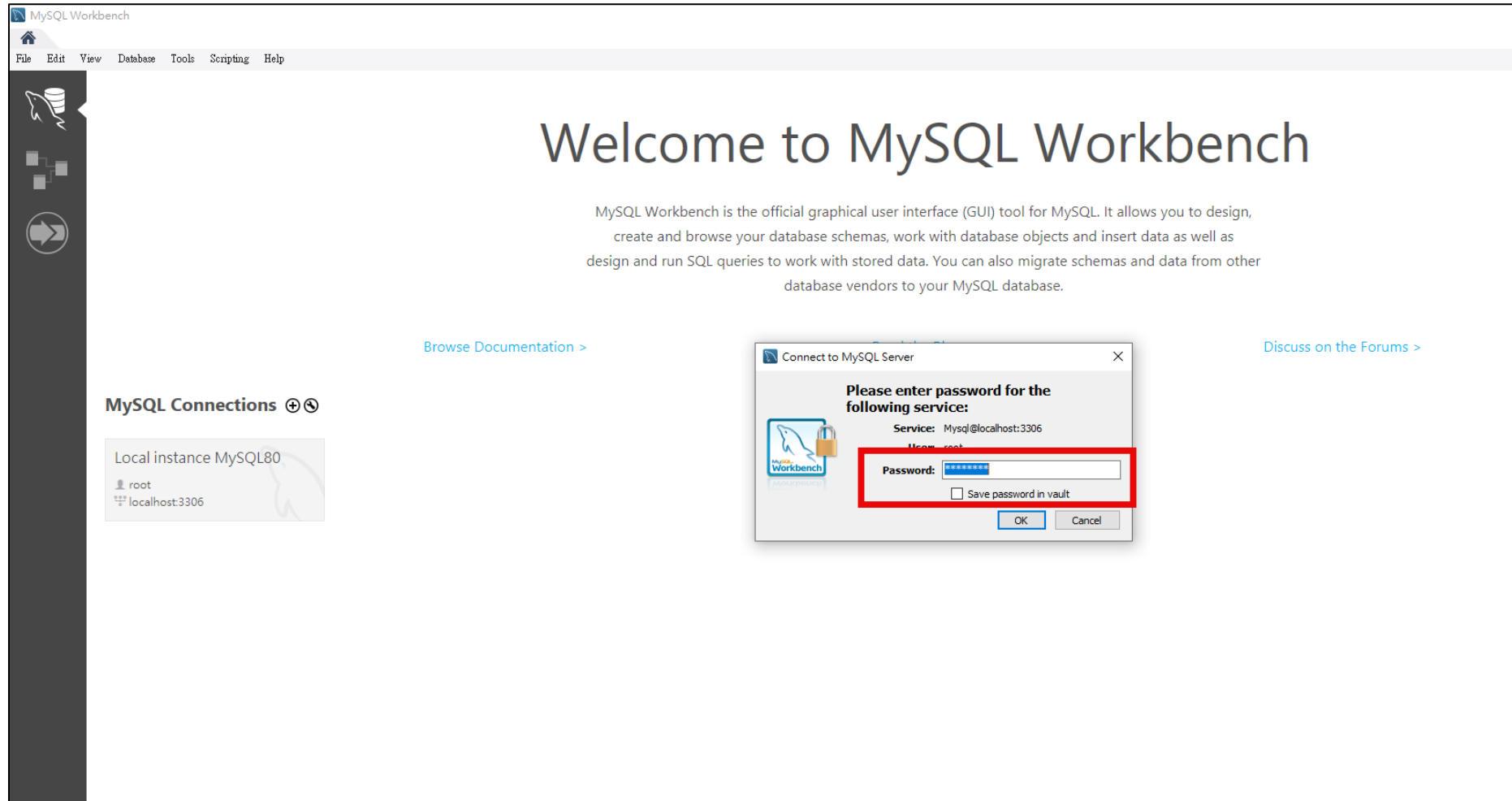
mysql> exit
```

# MySQL8.0安裝與設定(Installation & Settings)

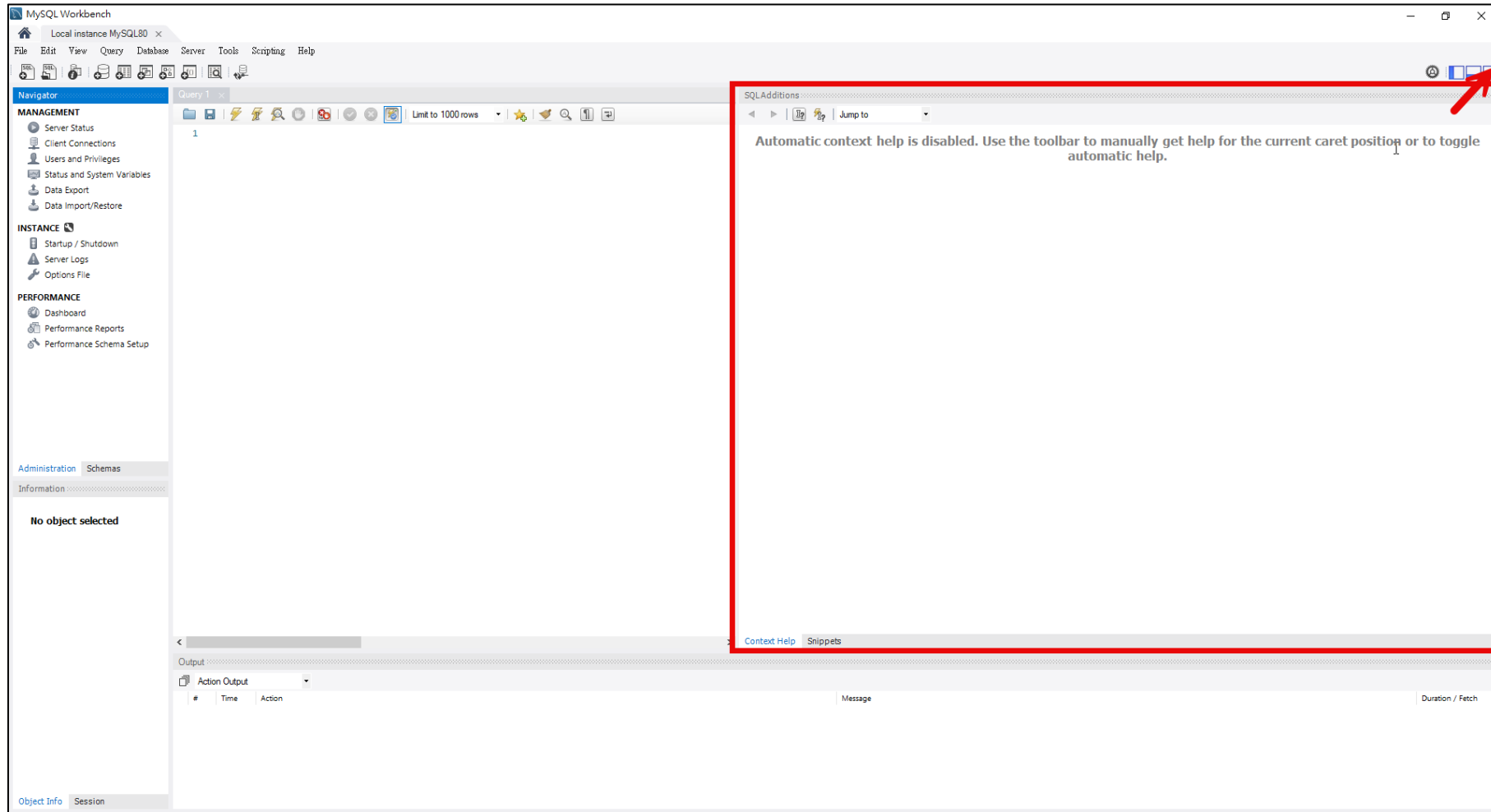
## ● Workbench介紹與設定



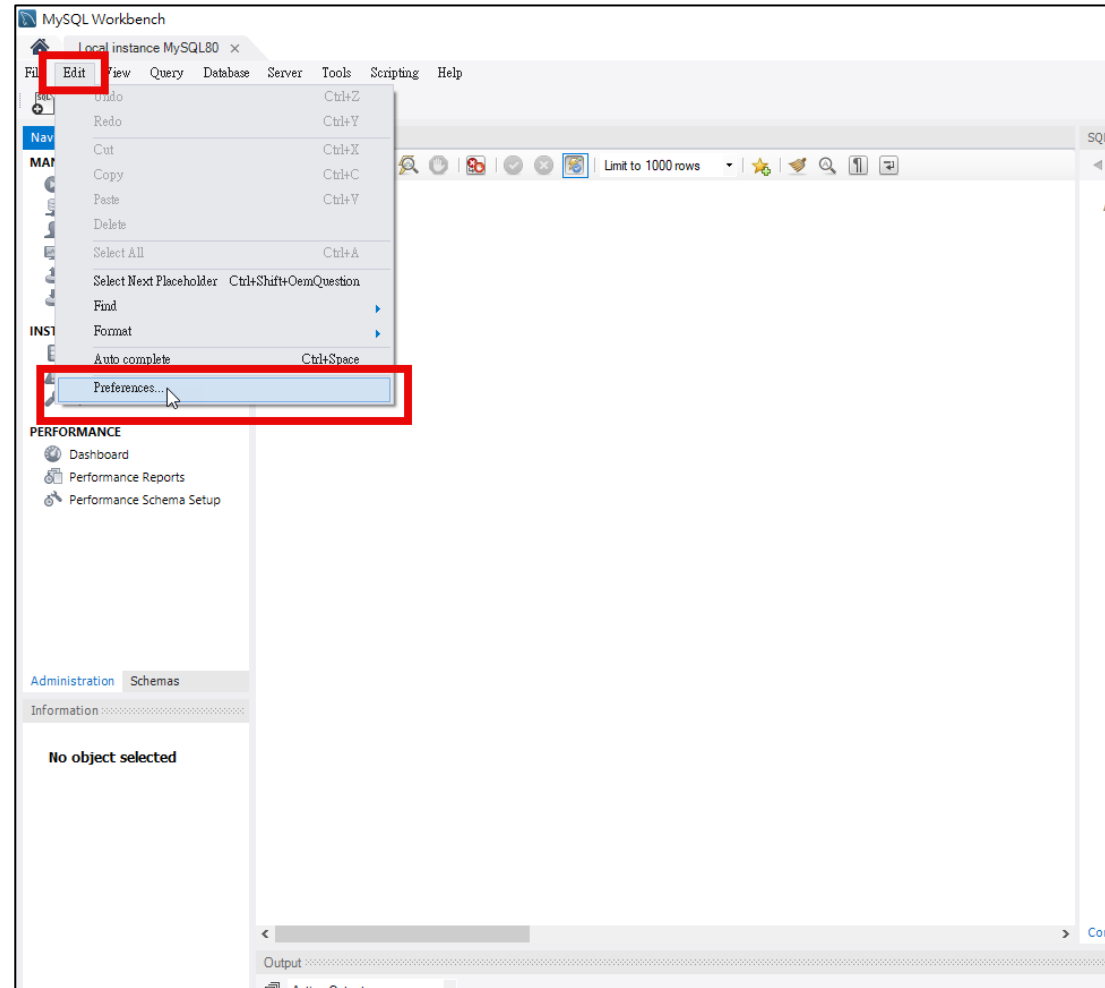
# MySQL8.0安裝與設定(Installation & Settings)



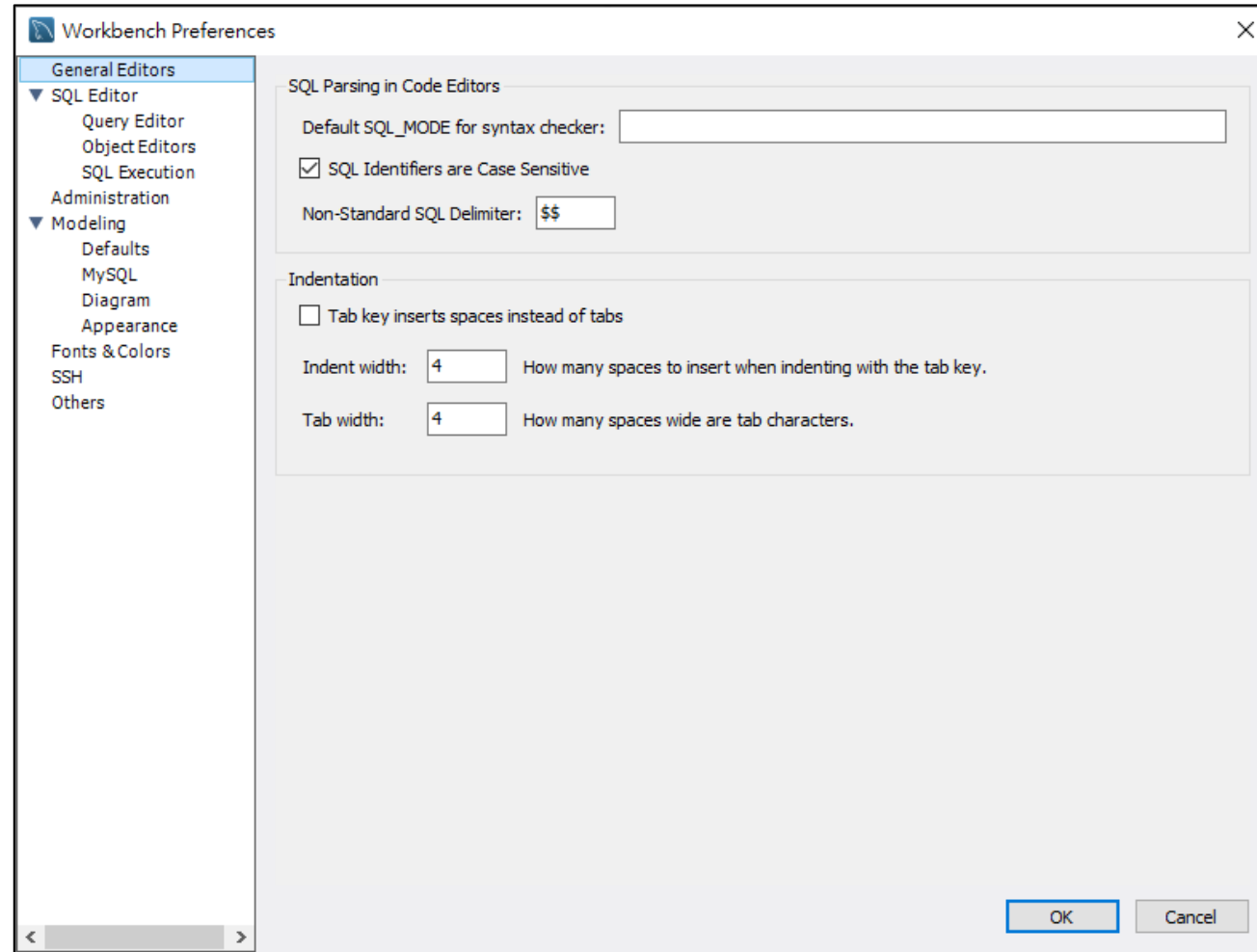
# MySQL8.0安裝與設定(Installation & Settings)



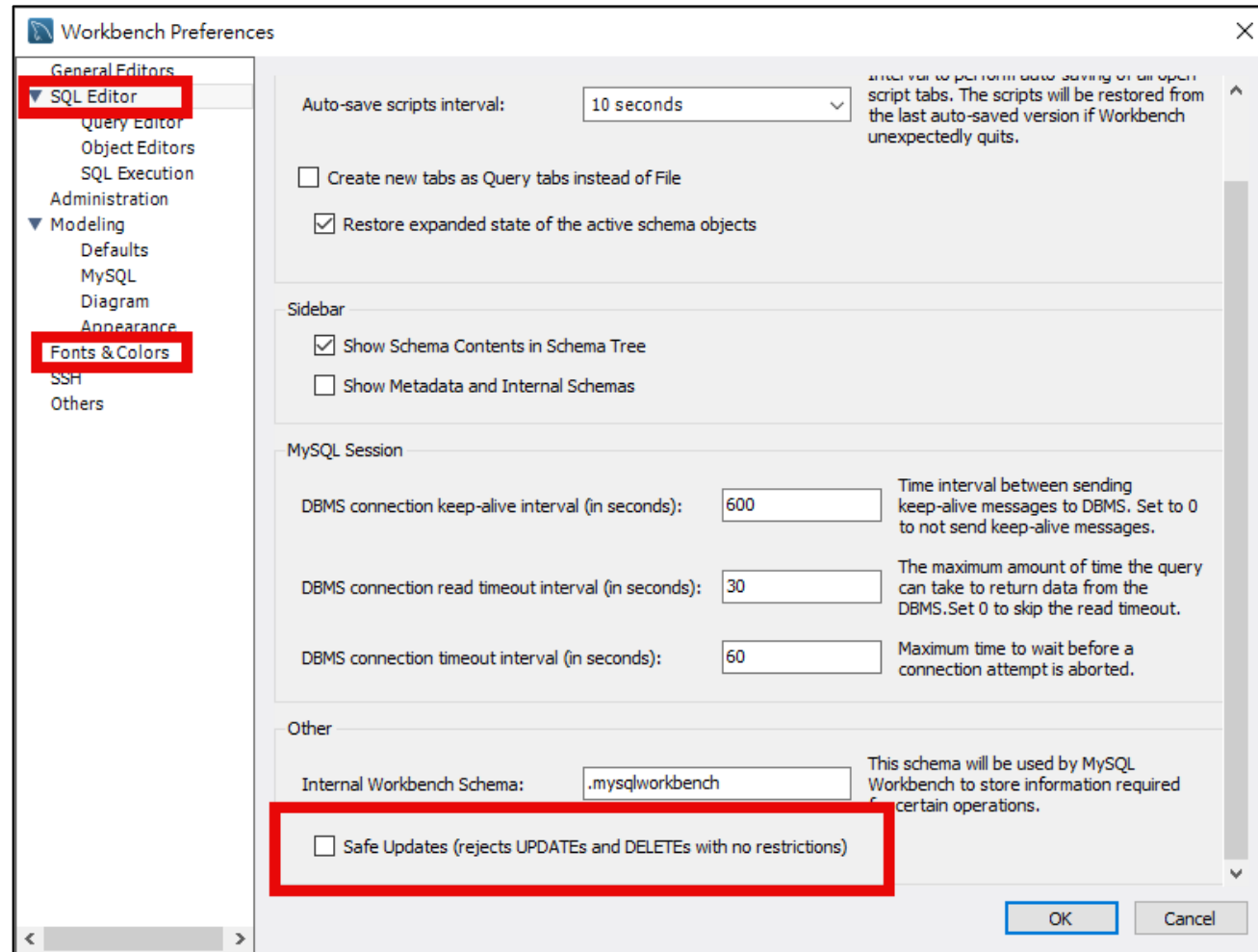
# MySQL8.0安裝與設定(Installation & Settings)



# MySQL8.0安裝與設定(Installation & Settings)



# MySQL8.0安裝與設定(Installation & Settings)



# 基本查詢(Basic Query)

# 基本查詢(Basic Query)

---

## ● 資料庫(Database)

- 資料庫管理系統(DBMS) → 資料庫(Database) → 表格(Table) → 資料(Data)  
檢視整個資料庫管理系統中有多少個database

```
show databases;
```

建立資料庫的語法

```
create database my_database;
```

刪除資料庫的語法

```
drop database my_database;
```

- 執行(Execute)分為

1. 選取區段執行或全部執行
2. 執行游標所在的區域(通常以 ; 劃分)



# 基本查詢(Basic Query)

---

- 使用資料庫

在資料庫管理系統中可能有多個資料庫，必須先選定欲使用的資料庫，這相當的重要，會建議養成常常檢查現在正在使用哪一個資料庫的好習慣。

```
use my_database;
```

檢視使用中資料庫的語法

```
select database();
```

# 基本查詢(Basic Query)

---

## ● 註解(Comment)

- MySQL資料庫的註解有兩種

### 1. 單行註解 -- 或 #

```
-- show databases;  
或  
# show databases;
```

### 2. 多行註解 /\* \*/

```
/*  
drop database my_database;  
drop table employee;  
*/
```

# 表格(Tables)

# 表格(Tables)

---

## ● 表格(Tables)

- 在資料庫中扮演重要的角色，資料庫中通常有若干個表格，同一個表格格式會一致。

表格中，橫的第一列稱之為 columns 或者 headers，也就是欄位名稱；下一列開始的每一列都稱之為一筆資料 row。

employee_id	employee_name	employee_age	employee_salary	employee_department
1	Tim	39	100000	Sales
2	Danny	27	33000	Accounting
3	Wilson	33	42000	Administration
4	Elizabeth	22	29000	Accounting

<< 假設情境 >> 主管問，可以用中文來命名表格或欄位嗎？他看不懂英文。

# 表格(Tables)

## ● 資料型別(Data Types)

- 在表格中同一欄位的資料型別必須一致，以利後續運算或比較，通常會在資料庫設計的階段就將表格各欄位的資料型別訂定好，中途新增或更改可能要負擔額外的成本及風險。

employee_id	employee_name	employee_age	employee_salary	employee_department
1	Tim	39	100000	Sales
2	Danny	27	33000	Accounting
3	Wilson	33	42000	Administration
4	Elizabeth	22	29000	Accounting

```
create table employee(  
employee_id int,  
employee_name varchar(50),  
employee_age int,  
employee_salary int,  
employee_department varchar(50)  
);
```

# 表格(Tables)

- 以下簡單介紹MySQL中常用的資料型別及分類--

## 1. 數字型別(Numeric Data Types)

資料型別	型別內容	補充
INT	整數，佔4 byte	範圍是 -2,147,483,648 ~ 2,147,483,647
TINYINT	整數，佔1 byte	範圍是 -128 ~ 127
SMALLINT	整數，佔2 byte	範圍是 -32,768 ~ 32,767
BIGINT	整數，佔8 byte	範圍是 -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
DECIMAL	定點數	DECIMAL(P, D)
FLOAT	浮點數，佔4 byte	
DOUBLE	浮點數，佔8 byte	
...		

# 表格(Tables)

---

## 2. 日期型別(Date and Time Types)

資料型別	型別內容	補充
DATE	僅有日期	1000-01-01 ~ 9999-12-31
DATETIME	日期與時間	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
TIMESTAMP	日期與時間	1970-01-01 00:00:01.000000 UTC ~ 2038-01-19- 03:14:07.999999 UTC
TIME	僅有時間	
YEAR	僅有年份	
...		

# 表格(Tables)

---

## 3. 文字型別(String Data Types)

資料型別	型別內容	補充
CHAR	固定長度	0 ~ 255 characters
VARCHAR	浮動長度	0 ~ 65535 characters
TINYTEXT		0 ~ 255 characters
TINYBLOB		0 ~ 255 bytes
TEXT		0 ~ 65535 characters
BLOB		0 ~ 65535 bytes
...		

資料庫建置者與使用者都應注意資料型別的問題，以免對產品或服務造成影響(Y2K, 2038問題)

# 表格(Tables)

---

## ● 新增表格(Create Table)

- 養成好習慣，在做任何資料操作以前，先確認自己有位於資料庫所在的位置使用資料庫

```
use my_database;
```

新增表格語法

```
create table members(  
  member_id int,  
  member_name varchar(50),  
  member_gender varchar(20),  
  member_age int  
);
```

<<提示一>> 在建立表格時，將想要建立的欄位讀出來，這有助於確認建立的過程中是否符合邏輯。

<<提示二>> 建立表格時，命名過程請避免使用停用字。

# 表格(Tables)

---

- 查看表格內的欄位配置及設定(三種寫法)

```
show columns from members;  
describe members;  
desc members;
```

## ● 刪除表格(Drop Table)

- 刪除表格的語法

```
drop table members;
```

# 表格(Tables)

---

- 練習
  - 請依據現有的資料製作這個表格(不需輸入資料)，並命名為player\_02
  - 查看這個表格內的欄位配置及設定
  - 刪除這個表格

player_id	player_name	player_age	player_salary	player_team
1	Damian	33	42000000	Milwaukee Bucks
2	Devin	27	33000000	Phoenix Suns
3	Paul	33	42000000	Los Angeles Clippers
4	Anthony	22	10000000	Minnesota Timberwolves

# 表格(Tables)

## ● 新增資料(Insert Data)

- 在members表格輸入一筆資料，輸入資料的順序需與欄位順序相同(請自行重建members表格)

```
insert into members(member_id, member_name, member_gender, member_age)
values(1, 'Abigail', 'Female', 15);
```

完成後檢視members表格

```
select * from members;
```

如果不按照設定的順序輸入資料，會產生錯誤

```
insert into members(member_id, member_name, member_gender, member_age)
values(1, 'Abigail', 'Female');
```

在MySQL中，可使用下列指令來檢視錯誤訊息

```
show warnings;
```

	Level	Code	Message
▶	Error	1136	Column count doesn't match value count at row 1

# 表格(Tables)

---

嘗試另一種情況，設定輸入欄位時不要把沒有資料的欄位寫進敘述

```
insert into members(member_id, member_name, member_gender)
values(1, 'Abigail', 'Female');
```

正常狀況下，未輸入資料的欄位會被自動給予null。

- 一次新增多筆資料

```
insert into members
(member_id, member_name, member_gender, member_age)
values
(2, 'Maria', 'Female', 21),
(3, 'Gregory', 'Male', 35),
(4, 'Scott', 'Male', 39);
```

一次新增多筆資料時，務必注意輸入欄位資料型別、欄位順序、逗號隔開以及分號結束等細節。

# 表格(Tables)

---

- 練習

使用一次新增多筆資料來完成下列名稱為employee的表格，並呈現結果。

employee_id	employee_name	employee_age	employee_salary	employee_department
1	Tim	39	100000	Sales
2	Danny	27	33000	Accounting
3	Wilson	33	42000	Administration
4	Elizabeth	22	29000	Accounting

# 空值與預設值(Null & Default)

# 空值與預設值(Null & Default)

## ● 空值(Null)

- 使用desc觀察表格設定

```
desc employee;
```

Field	Type	Null	Key	Default	Extra
employee_id	int	YES		NULL	
employee_name	varchar(50)	YES		NULL	
employee_age	int	YES		NULL	
employee_salary	int	YES		NULL	
employee_department	varchar(50)	YES		NULL	

MySQL Documentation對Null的定義：

### 9.1.7 NULL Values

The NULL value means “**no data**.” NULL can be written in any lettercase.

Be aware that the **NULL value is different from values such as 0 for numeric types or the empty string for string types.**

# 空值與預設值(Null & Default)

---

<<假設情境>>公司新進一位員工，但尚未敘薪，也不知道會分配到哪一部門，資料如何輸入？

```
insert into employee(employee_id, employee_name)
values(10, 'Harrison');
```

- 使用not null來設定表格不可填入null值

<<假設情境>>二手汽車行新進商品，輸入資料時肯定知道廠牌跟顏色，但可能還沒決定要賣多少錢，該如何建立表格？

```
create table cars(
car_brand varchar(50) not null,
car_color varchar(50) not null,
car_sale_price int
);
```

新進一台二手車，只有廠牌跟售價

```
insert into cars(car_brand, car_sale_price)
values('Luxgen', 550000);
```

# 空值與預設值(Null & Default)

---

出現錯誤的原因是car\_color必須要有值(設定not null時要給default，否則會變為null，不可使其無值)

Error 1364 Field 'car\_color' doesn't have a default value

回頭檢查cars表格

```
select * from cars;
```

再一次新增資料，這次使用一次新增多筆

```
insert into cars(car_brand, car_color, car_sale_price)
values
('Luxgen', 'blue', 400000),
('Ford', 'white');
```

還是出現錯誤，這次的錯誤原因是第二筆資料中，欄位數量(3個)與輸入值的數量(2個)不符

Error 1136 Column count doesn't match value count at row 2

# 空值與預設值(Null & Default)

---

修正輸入資料

```
insert into cars(car_brand, car_color, car_sale_price)
values
('Luxgen', 'blue', 200000),
('Ford', 'white', null);
```

最後再檢視cars表格，看是否有成功輸入

```
select * from cars;
```

# 空值與預設值(Null & Default)

---

## ● 預設值(Default)

- 將前述cars表格的設定做修改

```
create table cars_02(  
  car_brand varchar(50) not null default 'unknown',  
  car_color varchar(50) not null default 'unknown',  
  car_sale_price int default 50000  
);
```

雖然不能輸入null但會自動給default

嘗試輸入空資料看結果

```
insert into cars_02()  
values();
```

# 空值與預設值(Null & Default)

---

<<假設場景>>路邊一台車況很糟糕的車，只知道是黑色的，其餘未知，如何輸入資料？

```
insert into cars_02(car_color)
values('black');
```

## ● 設定預設值(Default)但允許輸入空值(Null)

- 將前述cars\_02表格的設定做修改

```
create table cars_03(
car_brand varchar(50) default 'unknown',
car_color varchar(50) default 'unknown',
car_sale_price int default 50000
);
```

檢查表格屬性設定

```
desc cars_03;
```

# 空值與預設值(Null & Default)

---

比較一下，資料不明的欄位就輸入null跟直接忽略不輸入值的差異

```
insert into cars_03(car_brand, car_color, car_sale_price)
values(null, 'black', null);
```

以及

```
insert into cars_03(car_color)
values('black');
```

# 空值與預設值(Null & Default)

---

練習：

依照前述設定，在cars\_03表格繼續做出下列結果。

car_brand	car_color	car_sale_price
NULL	black	NULL
unknown	black	50000
Toyota	NULL	50000
Honda	NULL	50000

思考一下並且比較上述三種設定的嚴謹程度？

# 主鍵(Primary Key)

# 主鍵(Primary Key)

## ● 主鍵(Primary Key)

- 試觀察下列表格並敘述其問題

car_brand	car_color	car_sale_price
Luxgen	white	200000
Luxgen	white	200000
Luxgen	white	200000
Luxgen	green	220000
Luxgen	green	220000
Luxgen	blue	230000
Luxgen	blue	230000

當同樣名稱資料(如商品、顧客姓名、評論內容等)有複數筆時，沒有辦法做出識別，稱之為資料缺乏識別性，這時會需要像是 id 這樣的欄位來幫助我們識別資料。主鍵的設置將有助於增加資料的鑑別度與可讀程度，一個表格中可設置複數個主鍵。

# 主鍵(Primary Key)

---

- 設置主鍵的語法

```
create table cars_04(  
  car_id int,  
  car_brand varchar(50),  
  car_color varchar(50),  
  car_sale_price int,  
  primary key(car_id)  
);
```

或

```
create table cars_04(  
  car_id int primary key,  
  car_brand varchar(50),  
  car_color varchar(50),  
  car_sale_price int  
);
```

# 主鍵(Primary Key)

---

輸入資料測試

```
insert into cars_04(car_id, car_brand)
values(1, 'Luxgen');
```

再輸入一次同樣的資料會發生什麼事？

Error 1062 Duplicate entry '1' for key 'cars\_04.PRIMARY'

再輸入一筆id為2的資料

```
insert into cars_04(car_id, car_brand)
values(2, 'Luxgen');
```

<<假設情境>> 如果新增資料時忘記給主鍵會發生什麼事情？

# 主鍵(Primary Key)

---

## ● 自動增值(Auto Increment)

- 可以藉由自動增值的功能使在新增下一筆資料時能夠自動增加主鍵值，避免需重複查看前面的資料而造成浪費時間或重工的問題，每個表格中只有一個為主鍵的欄位可以做這個設定，系統給的預設初始值為1，語法如下：

```
create table cars_05(  
  car_id int not null primary key auto_increment,  
  car_brand varchar(50),  
  car_color varchar(50),  
  car_sale_price int  
);
```

以car\_05表格為例，主鍵輸入的該行中，只要主鍵名稱(car\_id)與資料型別(int)的順序固定好就好，其餘設定(not null primary key auto\_increment等)的順序沒有規定。

# 主鍵(Primary Key)

---

新增一筆資料到car\_05表格

```
insert into cars_05(car_brand, car_color)
values('BMW', 'black');
```

auto\_increment注意事項：

<<假設情境>> 接手工作的人不知道car\_id有設定auto increment的狀況，會發生什麼事？

```
insert into cars_05(car_id, car_brand, car_color)
values (5566, 'Luxgen', 'red');
```

<<假設情境>> 前一位同仁回來接手這個表格，繼續新增資料時會發生什麼事？

```
insert into cars_05(car_brand, car_color)
values ('Luxgen', 'red');
```

# 主鍵(Primary Key)

---

- auto\_increment客製化

因應各種不同專案需求，有時主鍵的起始值不會從1開始，可如此設定：

```
create table cars_06(  
  car_id int not null primary key auto_increment,  
  car_brand varchar(50),  
  car_color varchar(50),  
  car_sale_price int  
);
```

將主鍵的起始值設定為101

```
alter table cars_06 auto_increment = 101;
```

並輸入一筆資料測試

```
insert into cars_06(car_brand, car_color, car_sale_price)  
values('Mazda', 'white', 300000);
```

# 主鍵(Primary Key)

---

## ● 獨特鍵(Unique Key)

- 同一個表格中，若除了主鍵以外也有其他欄位想要設為不可重複，可使用unique key來處理。

```
create table user_account(  
  user_id int primary key auto_increment,  
  user_name varchar(100) not null unique,  
  user_password varchar(100) not null  
);
```

思考一下，為何本表格中的user\_name不使用auto\_increment？

新增一筆資料

```
insert into user_account(user_name, user_password)  
values('ilove5566', '5566');
```

# 主鍵(Primary Key)

---

再次新增一筆同一個user\_name的資料

```
insert into user_account(user_name, user_password)
values('ilove5566', '5566');
```

Error 1062 Duplicate entry 'ilove5566' for key 'user\_account.user\_name'

改新增一筆不同user\_name的資料

```
insert into user_account(user_name, user_password)
values('ihate5566', '5566');
```

思考一下，為何user\_id為3而非2？

# 主鍵(Primary Key)

---

- 練習

製作一個叫做my\_product的表格，主鍵欄位為product\_id，並設定自動增加主鍵值，請從21開始。欄位product\_name輸入產品名稱、欄位product\_price請輸入產品價格，請輸入5筆資料。

# 增刪查改(CRUD)

# 增刪查改(CRUD)

---

- CRUD是SQL中基礎的資料操作，包含了新增(Create)、查詢(Read)、改正(Update)、刪除>Delete)
- 新增一個叫做player的表格

```
create table player(  
  player_id int not null primary key auto_increment,  
  player_name varchar(50),  
  player_position varchar(50),  
  player_age int  
);
```

# 增刪查改(CRUD)

---

- 在player表格中新增若干筆資料

```
insert into player(player_name, player_position, player_age)
values
('Bobby', 'INF', 23),
('Luis', 'P', 30),
('Jonah', 'C', 28),
('Framber', 'P', 30),
('Pete', 'P', 27),
('Matt', 'INF', 29),
('Corbin', 'OF', 23),
('Juan', 'OF', 28);
```

# 增刪查改(CRUD)

---

- 使用select檢視player表格，\* 代表所有的資料，欄位順序為預設輸入時的排序，養成時時檢視的好習慣

```
select * from player;
```

- 也可以只select部分欄位檢視，如只檢視player\_name欄位的資料

```
select player_name from player;
```

- 或只檢視player\_position欄位的資料

```
select player_position from player;
```

# 增刪查改(CRUD)

---

- 當然也可以同時檢視複數個欄位的資料(用逗號隔開)，且欄位順序可以調整

```
select player_name, player_position from player;
```

- 思考一下下列兩種資料的呈現方式

```
select player_position, player_age, player_name from player;
```

以及

```
select player_id, player_name, player_age from player;
```

<<假設情境>> 老闆要求查詢的資料是球員的「年齡」，哪一種顯示方式較為合適？

# 增刪查改(CRUD)

---

## ● 使用where子句設下條件做查詢

- 正常商業邏輯下，資料庫使用者查詢時不太可能每次都作全部查詢(select \* from table)，一來不易查看，二來當資料量過大時影響查詢速度，可依使用者需求設條件縮小查詢結果顯示的範圍
- 在player表格中找出年齡為23歲的球員

```
select * from player where player_age = 23;
```

得到player\_name為Bobby與Corbin的兩筆資料，搭配適當的欄位選擇可以得到更精簡的結果

```
select player_name, player_age from player where player_age = 23;
```

# 增刪查改(CRUD)

---

- 其他where子句的應用方式，如在player表格中找出年齡大於(等於)28歲的球員

```
select * from player where player_age > 28;  
select * from player where player_age >= 28;
```

- 在player表格中找出年齡小於(等於)28歲的球員

```
select * from player where player_age < 28;  
select * from player where player_age <= 28;
```

- 在player表格中找出名字為Corbin的球員

```
select * from player where player_name = 'Corbin';
```

注意資料型別，若資料型別為字串則要記得在前後加上單引號''

<<假設情境>>輸入時一時手滑，沒有輸入正確的大小寫時，會不會造成查詢錯誤？

# 增刪查改(CRUD)

---

- 使用兩個或以上的條件執行where子句查詢

```
select * from player where player_position = 'P';
```

加上and的第二個條件後...

```
select * from player where player_position = 'P' and player_age < 28;
```

<<假設情境>>若要查詢年齡在27歲~29歲之間(含)且守備位置為P的球員，要怎麼做？

```
select * from player where player_age between 27 and 29;  
select * from player where player_age between 27 and 29 and player_position  
= 'P';
```

查詢特定欄位兩值之間的資料可使用between 值1 and 值2(通常值2會比值1大)

# 增刪查改(CRUD)

- 練習：查詢出下列這三個結果

<提示一> 新增表格前注意總欄位數

<提示二> where & between

1.

grocery_name	grocery_category	grocery_reserves
Beef	Meat	13
Milk	Dairy	15
Spinach	Vegetables	20
Cheese	Dairy	5
Pork	Meat	8
Beer	Beverage	60
Cabbage	Vegetables	21
Lamb	Meat	16

2.

grocery_id	grocery_name
3	Spinach
4	Cheese
5	Pork
6	Beer
7	Cabbage
NULL	NULL

3.

grocery_id	grocery_name	grocery_category
1	Beef	Meat
5	Pork	Meat
8	Lamb	Meat
NULL	NULL	NULL

# 增刪查改(CRUD)

---

## ● Aliases別名

- 用較簡單易懂的名稱方便使用者檢視原先名稱較長或較複雜的欄位，先檢視player表格

```
select player_name, player_position, player_age from player;
```

<<假設情境>>教練看不懂英文欄位，怎麼辦？

```
select player_name as 姓名, player_position as 守位, player_age as 年齡  
from player;
```

需注意若as後欄位名稱有空格，就要用'將其包起來，否則會出現錯誤

```
select player_name as '球員 姓名', player_position as '守備 位置',  
player_age as '球員 年齡' from player;
```

- as可以省略

```
select player_name 姓名, player_position 守位, player_age 年齡 from player;
```

# 增刪查改(CRUD)

---

## ● Update改正(更新)

- 在update中，where子句的撰寫很重要，建議先找到資料確認其位置及相關欄位後再行更新，因為更新(與刪除)是不可復原的。

- 在player表格中更改守位名稱為P的為Pitcher

1. 先寫出where子句

```
select * from player where player_position = 'P';
```

2. 再將update敘述加入

```
update player set player_position = 'Pitcher'  
where player_position = 'P';
```

檢視player表格，確認守位名稱已成功更新

```
select * from player;
```

# 增刪查改(CRUD)

---

<<假設情境>>update時，手滑沒有加上where子句的結果？

(課程中建議使用drop table player;處理，重新建表)

- 在player表格中將Pete的守位名稱更改為OF

1. 先寫出where子句，將Pete的資料取出

```
select * from player where player_name = 'Pete';
```

2. 再將update敘述加入

```
update player set player_position = 'OF'  
where player_name = 'Pete';
```

檢視player表格，確認Pete的守位已成功更新為OF

```
select * from player;
```

# 增刪查改(CRUD)

---

- 練習：

1. 在player表格中，將Bobby的年齡改為26歲，並顯示此筆資料的名字與年齡欄位
2. 在player表格中，將Jonah的名字改為Jonathan，並與Bobby的資料一同顯示，欄位包含名字與年齡

<提示> 其他邏輯運算子

# 增刪查改(CRUD)

---

## ● Delete刪除

- 在delete中，也會搭配where使用，跟update一樣，一旦執行了就不可復原。
- Jonathan轉隊了，要將其資料刪除

1. 一樣，先寫出where子句

```
select * from player where player_name = 'Jonathan';
```

2. 再將delete敘述加入

```
delete from player  
where player_name = 'Jonathan';
```

檢視player表格，確認Jonathan的資料已成功刪除

```
select * from player;
```

# 增刪查改(CRUD)

---

<<假設情境>>delete時，手滑沒有加上where子句的結果？

(delete表格中所有資料並不同刪除表格，課程中建議使用drop table player;處理，重新建表)

- 在已被刪光資料的player表格中新增一筆資料，player\_id會從多少開始？

```
insert into player(player_name)
values('Justin');
```

<<假設情境>>多次的輸入又刪除後，如果想讓player\_id從1開始怎麼辦？

- 思考一下drop table vs. delete from

# 外來鍵(Foreign Key)

# 外來鍵(Foreign Key)

---

- 外來鍵是指在關聯式資料庫中，每一個資料表都是由資料間的關聯來連結彼此的關係。而主鍵會放在另一個資料表以建立彼此的關聯。

- 建議變更資料庫

```
create database social_media_app;
```

一樣，養成隨手use資料庫的好習慣

```
use social_media_app;
```

- 新增一個表格命名為users，注意id欄位要設定為auto\_increment

```
create table users(  
  id int not null primary key auto_increment,  
  user_name varchar(200)  
);
```

# 外來鍵(Foreign Key)

---

在users表格中輸入幾筆資料

```
insert into users(user_name)
values
('Amanda'), ('Brian'), ('Cally'), ('Daniel'), ('Edward');
```

- 另外新增一個表格命名為photos，並設定user\_id為外來鍵，**指向**users表格的id欄位，表示user\_id是參考user的id欄位，表達他們之間的關聯。

```
create table photos(
id int not null primary key auto_increment,
photo_url varchar(200),
user_id int,
foreign key (user_id) references users(id)
);
```

# 外來鍵(Foreign Key)

---

也在photos表格中輸入資料

```
insert into photos(photo_url, user_id)
values('https://123456.png', 1);
```

檢查photos表格

```
select * from photos;
```

再輸入幾筆資料

```
insert into photos(photo_url, user_id)
values
('https://111111.png', 1),
('https://222222.png', 1),
('https://333333.png', 1);
```

# 外來鍵(Foreign Key)

---

<<假設場景>>如果在photos表格輸入一筆不存在的使用者資料，會怎麼樣？

```
insert into photos(photo_url, user_id)
values('https://777777.png', 567);
```

Error 1452 Cannot add or update a child row: a **foreign key constraint** fails  
(`social\_media\_app`.`photos`, CONSTRAINT `photos\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`))

<<假設場景>>有一張不知道作者為何人的照片，如何輸入？

```
insert into photos(photo_url, user_id)
values('https://999999.png', null);
```

# 外來鍵(Foreign Key)

---

- 刪除的約束模式restrict, cascade, set null, no action。

- restrict

這種約束在表格建立之時就要設定，無法中途更改。

<<假設場景>> 若想要刪除users資料表中某個id的資料或整個資料表，嘗試執行之。

```
drop table users;  
delete from users where id = 1;
```

MySQL的預設就是restrict的刪除約束模式。

嘗試刪除整個表格

```
drop table photos;
```

# 外來鍵(Foreign Key)

---

- cascade

再新增一個表格命名為photos\_02

```
create table photos_02(  
  id int primary key auto_increment,  
  photo_url varchar(200),  
  user_id int,  
  foreign key(user_id) references users(id) on delete cascade  
);
```

輸入新的資料測試

```
insert into photos_02(photo_url, user_id)  
values  
( 'https://50v05C2qYeQBPfvV.png', 1), ( 'https://9939P61ncLk0zT71.png', 1),  
( 'https://IDiRYiItNd5TC2h9.png', 2), ( 'https://LsrdCdC0dhjrjteg.png', 2),  
( 'https://TKHN7Fnmeoepeahw.png', 3), ( 'https://ajG9183iiGYHoReq.png', 3),  
( 'https://edJKy1VdLkZ8wv5W.png', 4), ( 'https://nbiLUDgfcwI4ubWE.png', 4),  
( 'https://pfhednPD67rDnreQ.png', 5), ( 'https://ux1InX2oBS6YtBiG.png', 5);
```

# 外來鍵(Foreign Key)

---

將user表格中id為3的資料刪除

```
delete from users where id = 3;
```

再次檢視photos\_02資料表

```
select * from photos_02;
```

# 外來鍵(Foreign Key)

---

- set null

再新增一個表格命名為photos\_03

```
create table photos_03(  
  id int primary key auto_increment,  
  photo_url varchar(200),  
  user_id int,  
  foreign key(user_id) references users(id) on delete set null  
);
```

輸入資料測試(使用photos\_02的資料即可)

思考一下，這裡為何報錯？

# 外來鍵(Foreign Key)

---

將users表格中id為4的資料刪除

```
delete from users where id = 4;
```

檢查photos\_03表格

```
select * from photos_03;
```

- no action等同restrict

# 日期與時間(Date & Time)

# 日期與時間(Date & Time)

---

- 在表格中設定時間或日期相關的欄位

- 新增表格phone

```
create table phone(  
  phone_name varchar(30),  
  phone_price int,  
  stocking_time timestamp not null  
);
```

檢視欄位資訊並嘗試輸入資料

```
insert into phone(phone_name, phone_price, stocking_time)  
values  
( 'iphone 16 pro max 1TB', 60000, '2025-02-26 21:00:00');
```

# 日期與時間(Date & Time)

---

- 設定輸入新增資料當下的日期時間，可使用函數now()

```
create table phone_02(  
  phone_name varchar(30),  
  phone_price int,  
  stocking_time timestamp not null default now()  
);
```

輸入資料

```
insert into phone_02(phone_name, phone_price)  
values('iphone 16 pro max 1TB', 60000);
```

思考一下，選擇使用timestamp或datetime商業邏輯是什麼？

# 連結(Join)

# 連結(Join)

---

## ● Join能將資料庫中分屬兩個或以上不同表格的資料連接在一起做查詢

- 思考一下，外送平台的商業模式下，公司的資料庫裡可能有什麼表格？
  1. 使用者(顧客)資料：內含姓名、電話、住址、帳號、密碼、訂單等資料。
  2. 使用者(店家)資料：內含店名、餐廳種類、聯絡人、電話、店址、訂單等資料。
  3. 訂單資料(顧客)：內含訂單號碼、日期、餐廳、餐點名稱、金額、外送地址等資料。
  4. 收入資料：內含使用者付費金額、付給店家的抽成、公司自己的抽成等資料

.....

<<假設情境>>

想要查詢某使用者今年內曾經點過的所有餐點名稱，如何呈現？

想要查詢某店家的某筆訂單的抽成方式，如何呈現？

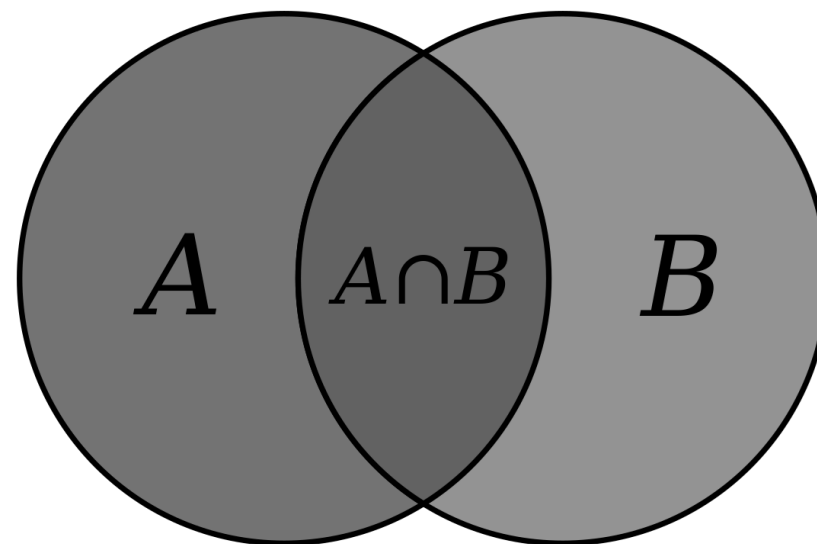
某顧客因為搬家，想一次修改其於使用者資料與訂單資料中的「地址」欄位，是否可能辦到？

# 連結(Join)

---

## ● Join的分類

- 大致上分為4種，通常視情況決定要如何使用
  1. (Inner) Join(一般的Join，最常使用)
  2. Left (Outer) Join
  3. Right (Outer) Join
  4. Full Join



# 連結(Join)

## ● Join的作法(Inner Join)

- 新增表格passengers, ports, class並輸入資料
- 將乘客姓名(pname)與登船碼頭的城市名(city)做對照，但兩者分屬不同表格，如何實現？

Join的邏輯，做之前可以嘗試讀出來，讓自己更了解現在自己執行到哪一步--

1. 想要顯示出的欄位填入select，可先用\*代替，待確認查詢無誤後再依需求填入需要的欄位
2. from後則填入主表格名稱
3. join後填入次表格名稱
4. on 主次兩表相同的欄位

```
select *  
from passengers  
join ports  
on portid = ports.id;
```

→

```
select pname, city  
from passengers  
join ports  
on portid = ports.id;
```

# 連結(Join)

---

若select欄位名稱在兩個表格中都有出現時會出現錯誤

```
select id, pname, city  
from passengers  
join ports  
on portid = ports.id;
```

Error Code: 1052. Column 'id' in field list is **ambiguous**

需在select時就註明清楚是要哪個表格的「id」，on敘述中雖沒有強制但建議也要標明清楚

```
select passengers.id, pname, portid, ports.id, city  
from passengers  
join ports  
on passengers.portid = ports.id;
```

雖然麻煩，但若能夠讓閱讀SQL程式碼的人避免不必要的誤會，就是更好的結果。

# 連結(Join)

<<假設情境>> 在顯示join結果時，要如何知道哪個id是哪個表的？

id	pname	id	city
1	Braund, Mr. Owen Harris	1	Southampton
3	Heikkinen, Miss. Laina	1	Southampton
4	Futrelle, Mrs. Jacques He...	1	Southampton
5	Allen, Mr. William Henry	1	Southampton

```
select pa.id as 乘客ID, pname, portid, po.id as 碼頭ID, city
from passengers as pa
Join ports as po
on portid = po.id;
```

若select的表格使用aliases，在on的地方也要使用aliases敘述。另，as其實可省略。

```
select pa.id 乘客ID, pname, portid, po.id 碼頭ID, city
from passengers pa
Join ports po
on portid = po.id;
```

# 連結(Join)

---

- 練習：

請利用passengers, class表格查詢出下表的結果

乘客姓名	船艙等級
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	First Class
Futrelle, Mrs. Jacques Heath (Lily May Peel)	First Class
McCarthy, Mr. Timothy J	First Class
Nasser, Mrs. Nicholas (Adele Achem)	Second Class
Braund, Mr. Owen Harris	Third Class
Heikkinen, Miss. Laina	Third Class
Allen, Mr. William Henry	Third Class
Moran, Mr. James	Third Class
Palsson, Master. Gosta Leonard	Third Class
Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	Third Class

# 連結(Join)

---

## ● Join with null(Left Join, Right Join, Full Join)

- 若在執行join時表格中有null值，則不會出現在join的結果內  
加入一筆沒有portid的乘客資料

```
insert into passengers(id, pclass, pname, sex, age, portid)
values(11, 2, 'Nancy Alomar', 'female', 54, null);
```

但再度執行join後卻沒有出現這筆資料

```
select *
from passengers
join ports
on portid = ports.id;
```

因為null沒有辦法符合on子句中的「等於」條件。

# 連結(Join)

- Left Join

Join時保留左邊表格的資料是完整的(含null值)

```
select *  
from passengers  
left join ports  
on portid = ports.id;
```

id	pclass	pname	sex	age	portId	id	embarked	city
8	3	Palsson, Master. Gosta Le...	male	2	1	1	S	Southampton
9	3	Johnson, Mrs. Oscar W (El...	female	27	1	1	S	Southampton
10	2	Nasser, Mrs. Nicholas (Ad...	female	14	2	2	C	Cherbourg
11	2	Nancy Alomar	female	54	NULL	NULL	NULL	NULL

保留了左方表格含null的資料(沒有portid的乘客資料)出現，且對應之ports表格資料都給null

# 連結(Join)

- Right Join

Join時保留右邊表格的資料是完整的(含null值)，先手動加入一筆沒有embarked的資料到ports表格

```
insert into ports(id, embarked, city)
values(4, null, 'Liverpool');
```

執行right join

```
select *
from passengers
right join ports
on portid = ports.id;
```

id	pclass	pname	sex	age	portId	id	embarked	city
2	1	Cumings, Mrs. John Bradl...	female	38	2	2	C	Cherbourg
10	2	Nasser, Mrs. Nicholas (Ad...	female	14	2	2	C	Cherbourg
6	3	Moran, Mr. James	male	NULL	3	3	O	Queenstown
NULL	NULL	NULL	NULL	NULL	NULL	4	NULL	Liverpool

保留了右方表格含null的資料(沒有embarked的碼頭資料)出現，且對應之passengers表格資料都給null

<<假設情境>> 執行left join或right join時，若互換表格順序會有什麼影響？

# 連結(Join)

---

- Full Join

Join時保留兩邊表格的資料是完整的(含null值)

執行full join

```
select *  
from passengers  
full join ports  
on portid = ports.id;
```

毫無反應，就只是個join

MySQL目前尚無full join，

故需使用union語法來取代

→

```
select *  
from passengers  
left join ports  
on portid = ports.id  
union  
select *  
from passengers  
right join ports  
on portid = ports.id;
```

# 連結(Join)

---

- Join加上where子句以設定其他條件  
請找出所有女性乘客的姓名，以及她們登船的城市名稱

```
select pname, sex, city
from passengers
join ports
on portid = ports.id
where sex = 'female';
```

- 練習：  
請找出所有乘坐Second Class與Third Class客艙乘客的姓名，以及該客艙的class\_level

<提示一>總共應有8筆資料

<提示二>pclass

# 連結(Join)

## ● 三個表格的join(Three Tables Join)

- 練習

teams表格

id	team_name	team_nickname
1	Brother Elephants	爪爪
2	Rakuten Monkeys	吱吱
3	Uni Lions	喵喵
4	Fubon Guardians	邦邦
5	WeiChuan Dragons	油龍
NULL	NULL	NULL

result表格

id	result_name	result_description
1	贏球	得分比對方多
2	輸球	得分比對方少
3	平手	雙方得分一樣多
NULL	NULL	NULL

matches表格

fk_result_id	fk_team_id
1	2
1	3
1	5
1	4
2	1
2	2
2	4
2	5
3	1
3	3

# 連結(Join)

請做出下列輸出結果

team_nickname	result_name
吱吱	贏球
喵喵	贏球
油龍	贏球
邦邦	贏球
爪爪	輸球
吱吱	輸球
邦邦	輸球
油龍	輸球
爪爪	平手
喵喵	平手

思考一下這個輸出結果要如何做？

team_nickname	result_name
爪爪	輸球
爪爪	平手
吱吱	贏球
吱吱	輸球
喵喵	贏球
喵喵	平手
邦邦	贏球
邦邦	輸球
油龍	贏球
油龍	輸球

# 分組與聚成(Group By & Aggregation)

# 分組與聚成(Group By & Aggregation)

---

## ● 分組與聚成(Group by & Aggregation)

- 將多筆資料根據特定欄位的規則分組，方便呈現

```
select * from passengers  
group by age;
```

發現意義不大，需與聚成函數搭配使用

分組呈現各等級船艙內人數

```
select count(pclass) from passengers  
group by pclass;
```

分組呈現各性別乘客人數

```
select count(sex) from passengers  
group by sex;
```

# 分組與聚成(Group By & Aggregation)

---

- 聚成函數種類
  1. count()：計算分組內資料個數總和
  2. sum()：計算分組內資料總和
  3. avg()：計算分組內的平均數
  4. max()：計算分組中的最大值
  5. min()：計算分組中的最小值
- 練習
  1. 分組呈現各性別的平均年齡
  2. 分組呈現各船艙等級的最大年齡歲數
  3. 分組呈現各船艙等級的最小年齡歲數

# 分組與聚成(Group By & Aggregation)

## ● 分組與連結

- 當想要計算的分組分處不同的表格，就必須加上join  
artist表格

id	artist_name
1	Bruno Mars
2	Jay Sean
4	Sean Kingston
3	Usher
NULL	NULL

song表格

id	song_name	artist_id
1	Just the way you are	1
2	Treasure	1
3	Down	2
4	Yeah	3
5	DJ got us fall in love again	3
6	Beautiful girls	4
NULL	NULL	NULL

→

預期呈現結果：每位歌手有多少首歌

artist_name	numbers of songs
Bruno Mars	2
Jay Sean	1
Sean Kingston	1
Usher	2

# 分組與聚成(Group By & Aggregation)

---

## ● 分組中條件(Having)

- 承上題，若想要找出2首歌以上的歌手，可以再次設條件查詢，使用having

```
select artist_name, count(artist_id) 'numbers of songs'
from artists
join songs
on artist_id = artists.id
group by artist_name
having count(*) > 1;
```

- 練習：在passengers與ports表格中找出搭乘人數大於100人以上的城市名稱以及搭乘人數(欄位名稱請使用city, boarding counts)

# 排序(Order by)

# 排序(Order by)

---

## ● 排序(Order by)

- 資料的呈現會依據商業需求而有所不同，可以使用order by來排序資料想呈現的方式  
將乘客資料依照姓名字母順序列出

```
select * from passengers  
order by name;
```

將乘客資料的排序依照年齡列出，預設為由小到大

```
select * from passengers  
order by age;
```

將乘客資料的排序依照年齡由大到小列出

```
select * from passengers  
order by age desc;
```

# 排序(Order by)

---

- 練習：在passengers表格將乘客依照portid昇冪排列，並將null值置於表格最後。

# 限制與跳過(Limit & Offset)

# 限制與跳過(Limit & Offset)

---

## ● 限制與跳過(Limit & Offset)

- 選取資料的前若干筆

```
select * from passengers limit 10;
```

就算選取超過資料筆數的上限，也只會回傳最多的資料筆數，不會報錯

```
select * from passengers limit 1000;
```

- [index, count]用法或搭配offset以間隔查詢區段  
從第index筆開始，回傳count筆資料(Index從0開始)

```
select * from passengers limit 5, 20;
```

搭配offset

```
select * from passengers limit 20 offset 5;
```

# 限制與跳過(Limit & Offset)

---

- 練習：請從乘客資料中依照portid由小至大找出前15筆資料

# 子查詢(Subquery)

# 子查詢(Subquery)

---

## ● 子查詢(Subquery)

- 查詢中的查詢，子查詢要使用小括號包起來，資料庫會先查詢小括號中的查詢，再執行外層的。  
請找出所有乘客中，年紀比二等客艙最年長者還要老的乘客，列出其客艙等級、姓名與年齡

1. 找出二等客艙最年長者的年紀

```
select max(age) from passengers where pclass = 2;
```

2. 找出所有乘客中年紀比二等客艙最年長者還要老的乘客

```
select * from passengers where age > 70;
```

3. 將找出的年齡如何查詢到的語法帶入，並寫出要顯示的欄位名稱(合併前兩個select)

```
select pclass, name, age  
from passengers where age > (select max(age) from passengers where pclass = 2);
```

# 區別(Distinct)

# 區別(Distinct)

---

## ● 區別(Distinct)

- 查詢時將重複的值去除，可運用於快速尋找欄位中有什麼內容  
只找出有哪些pclass，重複的不要

```
select distinct pclass from passengers;
```

只找出有哪些portid，重複的不要

```
select distinct portid from passengers;
```

練習：搭配聚成函數，算出總共有幾個等級的客艙，並給予欄位名稱class\_count

# 區別(Distinct)

---

- 雙重distinct，可設置兩個distinct條件

```
select distinct pclass, sex from passengers;
```

同樣可以搭配聚成函數

```
select count(distinct pclass, sex) from passengers;
```

# 模糊搜尋(Like)

# 模糊搜尋(Like)

---

## ● 模糊搜尋(Like)

- 查詢表格中可能有出現的內容，適合查詢文字資料型別的資料  
可搜尋部分字串，與where比較起來較寬鬆  
尋找乘客姓名中有william的乘客

```
select * from passengers where name like '%william%';
```

思考一下，如果要的只是名字是william的乘客，要怎麼做？

```
select * from passengers where name like '% william %';  
select * from passengers where name like '% william %' and sex = 'male';
```

比較一下上述兩種寫法，思考不同商業邏輯下的查詢方式

<<假設情境>> 想要查詢姓氏為Williams的乘客名單，怎麼做？

# 模糊搜尋(Like)

---

- 單偏%(Wildcard)

尋找姓氏為smith的乘客名單

```
select * from passengers where name like 'smith%';
```

尋找姓名欄位中有註解的乘客名單

```
select * from passengers where name like '%)';
```

可同時查詢兩種條件

```
select * from passengers where name like '%williams%' and name like '%charles%';
```

- 搜尋資料位數

```
select * from passengers where ticketid like '___';
```

# 模糊搜尋(Like)

---

- 練習：找出船票號碼是二位數，名字叫做william的乘客。

<<提示>>應為7筆

在內與非在內(In & Not In)

# 在內與非在內(In & Not In)

---

## ● 在內與非在內(In & Not In)

- 查詢表格中含或非含某條件的資料

尋找1號登船點與3號登船點上船的所有乘客名單

```
select * from passengers where portid = 1 or portid = 3;  
select * from passengers where portid in (1, 3);
```

尋找不是1號登船點與3號登船點上船的所有乘客名單

```
select * from passengers where portid != 1 and portid != 3;  
select * from passengers where portid not in (1, 3);
```

思考一下上述查詢中or與and的使用

練習：請查詢出所有不是女性且非從2號及3號登船點上船的乘客名單

# 關鍵字(Case)

# 關鍵字(Case)

---

## ● 關鍵字(Case)

- 替查詢設定條件並回傳值，類似程式中if/else的用法  
設條件查詢旅客的登船點並回傳設定的值

```
select id, name,  
case  
  when portid = 1 then 'Southampton'  
  when portid = 2 then 'Cherbourg'  
  when portid = 3 then 'Queenstown'  
  else 'unknown'  
end boarding_place  
from passengers;
```

# 變數(Variables)

# 變數(Variables)

---

## ● 變數(Variables)

- 分為自訂變數、區域變數、系統變數

宣告自訂變數

```
select @myname := 'Masataka Oniwa';
```

檢查設定結果

```
select @myname;
```

當然也可以一次設定多個自訂變數

```
select @myname := 'Sekiro', @myhome := 'Ashina';
```

# 變數(Variables)

---

- 給予變數值

```
set @myname := 'Tim', @myhome = 'Taipei';
```

檢查設定結果

```
select @myname, @myhome;
```

變數可運用於給予時間值

```
set @mytimestamp = now();  
select @mytimestamp;
```

另一種給予當下時間值的方法

```
set @mytimestamp = current_timestamp;
```

# 預儲程序(Stored Procedure)

# 預儲程序(Stored Procedure)

---

## ● 預儲程序的用法

- 類似設定函數，將重複性高或繁複的查詢內容儲存成一個特定名稱，需要使用時再用call呼叫此預儲程序，可以先儲存變數、運算等等。

使用begin...end來存放一個block

```
delimiter //  
create procedure select_ports()  
begin  
    select * from ports;  
end //  
delimiter ;
```

//才是真正的結束且執行，結束後記得換回；否則容易混淆設定

# 預儲程序(Stored Procedure)

---

- 使用call執行創建好的預儲程序

```
call select_ports();
```

- 刪除方式

```
drop procedure select_ports;
```

- 預儲程序的輸入參數

製作一個stored procedure，輸入portid就可以知道是哪個城市

```
delimiter //  
create procedure ports_city(in portsid int)  
begin  
    select city from ports where id = portsid;  
end //  
delimiter ;
```

```
call ports_city(3);
```

# 預儲程序(Stored Procedure)

---

- 練習：製作一個預儲程序，輸入乘客姓名(pname)的一部份就可以知道有這個姓名的人是哪個登船點登船的。

<提示一> pname

<提示二> like, wildcard

# 交易(Transaction)

# 交易(Transaction)

---

## ● 交易(Transaction)

- 確保資料庫的運行過程中，資料不遺失與保持資料完整性的機制，各大RDBMS皆會遵守交易機制的原理，交易具有以下四個特性
  1. 原子性(Atomicity)：意指整個交易只有全部成功，否則就整個交易取消，不會有執行到一半的情形。
  2. 一致性(Consistency)：意指交易完成後其資料庫的更新內容都完整及正確，且符合各原始設定中的條件。
  3. 隔離性(Isolation)：意指在交易的過程中所會用到的資料會各自獨立，不會造成各資料間互相影響，這部分可透過交易鎖定的方式來避免。
  4. 永久性(Durability)：意指整個交易完成且確認(Commit)後，受到影響的資料將被永久改變，不會因為外在因素(如系統問題或其他人為操縱)而將資料回復(Rollback)。

交易相關語法

```
start transaction;  
rollback;  
commit;
```

# 交易(Transaction)

---

- 自動認可交易模式(Autocommit Transactions Mode)為MySQL預設之交易模式，可使用下列語法手動關閉或開啟

```
set @@autocommit = off;  
set @@autocommit = on;
```

# 交易(Transaction)

---

交易練習：以銀行交易時遇到系統斷電為例。

試用表格

```
create database transaction_test;  
use transaction_test;
```

```
create table transaction_test(  
id int not null primary key auto_increment,  
user_name varchar(50),  
user_deposit bigint  
);
```

```
insert into transaction_test(user_name, user_deposit)  
values  
('Sam', 200000), ('Evelyn', 80000);
```

# 資料庫正規化(Database Normalization)

# 資料庫正規化(Database Normalization)

---

## ● 資料庫正規化(Database Normalization)

- 關聯式資料庫可透過正規化來更改資料表的結構提升資料庫效能，成為一個有效率的資料庫，其目的是使減少資料庫中重複的資料，以達到可快速搜尋資料的功能，這在資料量龐大的資料庫中會特別明顯，這些正規化的進流程是必須按照順序來的，一個階段完成才能進行下一個階段。於此介紹三階段的資料庫正規化。
  1. 第一階正規化(1st Normal Form: 1NF)：表格內的欄位只能存放單一值，且值不可為重複群體，需設立主鍵(Primary Key)。
  2. 第二階正規化(2nd Normal Form: 2NF)：符合第一階正規化後，將與主鍵無部份相依的欄位獨立出去，建立新表格，建立新主鍵，再用外來鍵(Foreign Key)與其他表格建立關係。如果一個資料表的主鍵只有單個欄位的話，即符合第二階正規化。
  3. 第三階正規化(3rd Normal Form: 3NF)：符合第二階正規化後，非主鍵屬性之間應該是獨立無關的，且不存在遞移關係，即欄位間不應有變更而互相影響。

# 資料庫正規化(Database Normalization)

---

- 通用的資料庫設計最多滿足第三階正規化即可，有一種說法是，若資料庫的正規化過高，雖然會有對資料間關係更好的約束性，但也可能讓表格數量增加而令資料庫的運作更加繁忙，進而影響效能。老話一句，視商業邏輯及使用者需求再來訂定適合的正規化流程。