

## 第十七組專題報告

### 一、專題說明

這次的專題我選的題目是英文補習班的後台管理系統，出發點是建立於我本人在英文補習班打工四年的經歷。該補習班目前紀錄及查詢學生繳費以及查找聯絡人都還在用紙本的方式，在現在這個電腦普及的時代感覺不太實際，我一直都很想做一個資料庫來實現歸納整理或是快速查詢的動作。

因此這次期末專題我設計了一套專屬於英文補習班的後台資料庫管理系統，目標是簡化行政流程、提升資料一致性與查詢效率。這個系統涵蓋了學生及老師的基本資料、學生緊急聯絡人資料、各個班級資訊、學生繳費紀錄、老師授課安排、老師教學時數等核心模組，同時支援學生報名課程、老師薪資計算與家長聯絡等常見情境。

資料庫設計方面，依據實務需求進行實體-關係模型（ERD）建構，並透過正規化程序使資料庫達到第三正規形式（3NF），以避免資料重複與異常。

進一步，此系統實作了多項進階 SQL 功能，例如：

- 利用 **Stored Procedure** 自動計算教師薪資
- 使用索引強化查詢效能
- 設計 **Trigger** 自動記錄異動行為
- 執行交易控制（**Transaction**）以保護資料一致性

本系統結合 **DBeaver** 開發環境與 **Node.js** 前端互動設計，展現資料庫與應用層整合之效能，最終以一套具備完整功能的補習班資訊系統作為成果展示。

## 二、專題實作內容

### 1. 資料庫需求分析與規劃

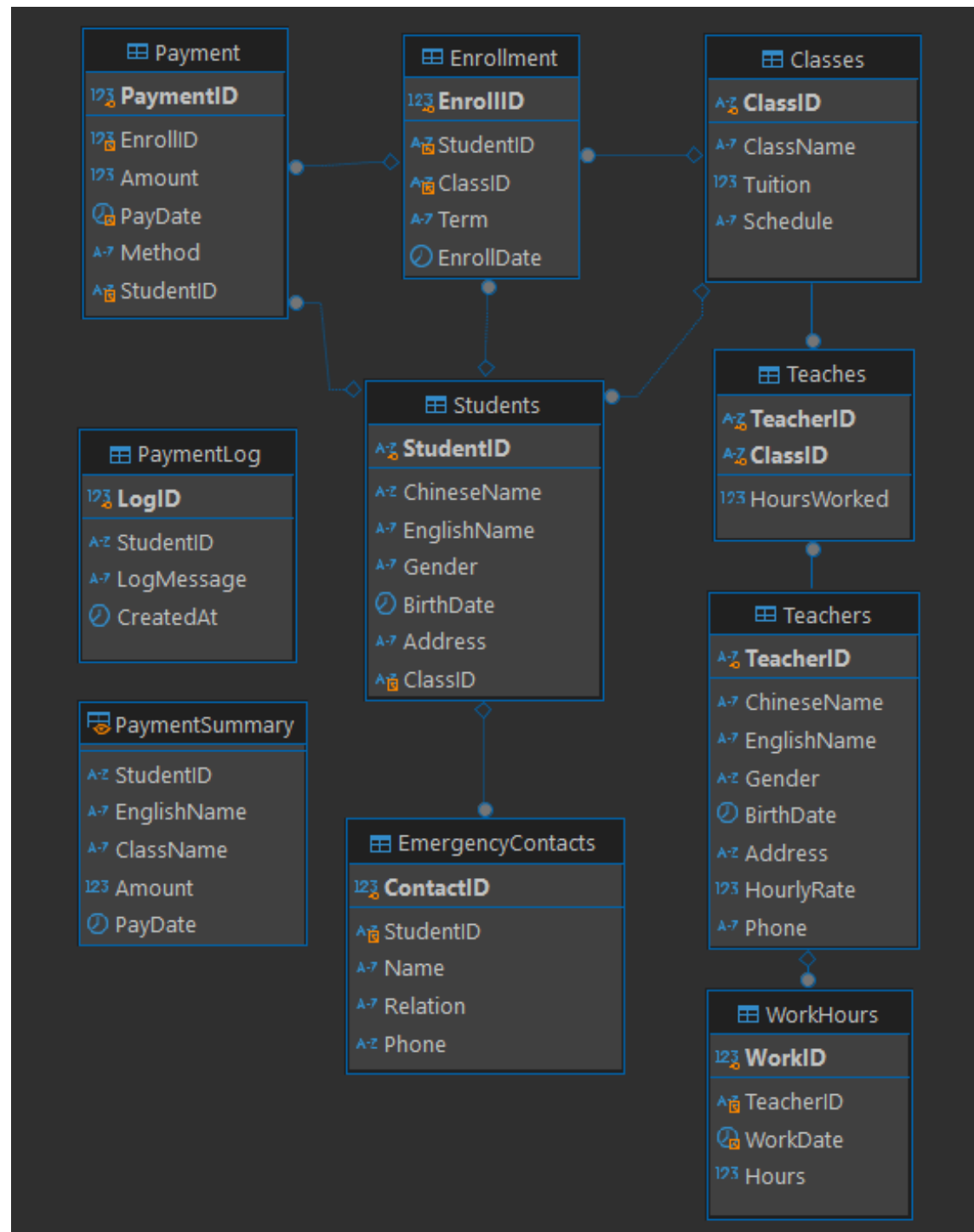
#### 系統功能構想：

本系統以補習班行政管理為核心，設計功能涵蓋學生、教師、班級、繳費與排課等資料之整合與應用。主要功能包括：

- **學生報名與管理：**學生可選擇報名既有班級，系統自動記錄報名日期與就讀班級，並保留學生基本資料與緊急聯絡人資訊。
- **課程與班級管理：**班級包含名稱、上課時段與對應學費，系統支援查詢每班學生名單與授課教師。
- **教師排課與時數統計：**可紀錄教師所授課班級與每日工作時數，並自動計算當月薪資。
- **繳費管理：**系統記錄學生每期繳費資料，提供報表查詢與逾期提醒。
- **資料查詢與報表輸出：**提供各式複雜查詢功能，例如：某教師授課班級、學生繳費歷史、熱門班級、薪資計算結果等。

透過資料庫層級的規劃與優化，本系統能有效支援補習班日常營運所需之資料作業流程。

## ER Diagram :



**a. ER 圖說明:**

Students (學生)	主鍵：StudentID 屬性：ChineseName, EnglishName, Gender, BirthDate, Address, ClassID
Teachers (老師)	主鍵：TeacherID 屬性：ChineseName, EnglishName, Gender, BirthDate, Address, HourlyRate, Phone
Classes (班級)	主鍵：ClassID 屬性：ClassName, Tuition, Schedule
Enrollment (選課紀錄)	主鍵：EnrollID 外鍵：StudentID, ClassID 屬性：Term, EnrollDate
Payment (繳費紀錄)	主鍵：PaymentID 外鍵：EnrollID, StudentID 屬性：PayDate, Amount, Method
TeacherClass (老師授課班級關聯)	複合主鍵：TeacherID, ClassID 屬性：HoursWorked (紀錄某老師對某班本月總教學時數)
WorkHours (教師上班時數)	主鍵：WorkID 外鍵：TeacherID 屬性：WorkDate, Hours
EmergencyContacts (緊急聯絡人)	主鍵：ContactID 外鍵：StudentID 屬性：Name, Relation, Phone
PaymentLog (繳費異動記錄)	主鍵：LogID 外鍵：StudentID 屬性：LogMessage, CreatedAt
PaymentSummary (繳費摘要報表)(View 或暫存表)	欄位：StudentID, EnglishName, ClassName, Amount, PayDate

## 正規化：

### **a. 未正規化前的資料:**

學生 S001，中文名「吳品霏」、英文名「Nova」，女性，生日 2007/01/23，地址「新北市蘆洲區中正路 123 號 5 樓」，就讀班級 C01「Puppy」，學費 9000 元，每週二、五 16:30-18:30，授課老師 T003（陳筠潔 Amy），女性，生日 1992/03/28，住在台北市大同區承德路五段 88 號，時薪 700 元，電話 0933444555；她在 2025 春季學期報名，報名編號為 E001，報名日期為 2025/02/01，付款紀錄為 P001，於 2025/02/10 繳費 9000 元，使用現金支付；緊急聯絡人為「吳媽媽」，關係為「母親」，電話 0912123456。

學生 S002，中文名「蘇宣羽」、英文名「Ivy」，女性，生日 2007/07/18，地址「新北市蘆洲區光華路 71 號 8 樓」，就讀班級 C01「Puppy」，學費 9000 元，每週二、五 16:30-18:30，授課老師 T003（陳筠潔 Amy），女性，生日 1992/03/28，住在台北市大同區承德路五段 88 號，時薪 700 元，電話 0933444555；她在 2025 春季學期報名，報名編號為 E002，報名日期為 2025/02/01，付款紀錄為 P002，於 2025/02/12 繳費 9000 元，使用轉帳支付；緊急聯絡人一為「蘇先生」，關係為「父親」，電話 0923456789，緊急聯絡人二為「洪小姐」，關係為「母親」，電話 0911222333。

(以下略)

---

### **b. 從 1NF 到 2NF:**

在初始 1NF 設計中，我們將所有學生、班級、老師、繳費、緊急聯絡人等資料整合在同一張大表中，雖然已滿足「每個欄位只能有一個值」的基本規範，但仍存在大量重複資訊（例如老師資料在多筆學生中反覆出現）、多重主鍵依賴、以及部分依賴問題。

#### **主鍵分析：**

在 1NF 階段，我們若以 (StudentID, ClassID) 作為主鍵，會發現部分欄位（如學生地址、生日）只依賴 StudentID，老師姓名、電話則依賴於 ClassID 或 TeacherID，這造成部分依賴（partial dependency），不符合 2NF 要求。

轉換為 2NF 的處理方式如下：

拆分「實體」成多張表格：

Students	以 StudentID 為主鍵，獨立儲存學生個人資訊（如姓名、性別、地址、生日）
Classes	以 ClassID 為主鍵，紀錄班級名稱、學費、時段
Teachers	以 TeacherID 為主鍵，儲存老師個資、時薪等
Enrollment	用來紀錄學生報名的班級，主鍵為 EnrollmentID，並包含 StudentID、ClassID
Payment	與報名資料一對一對應，用來記錄繳費金額、日期、方式
EmergencyContacts	每筆資料對應一位學生，記錄聯絡人姓名、關係、電話

建立外鍵來維護實體關係：

Enrollment.StudentID → Students.StudentID

Enrollment.ClassID → Classes.ClassID

Payment.EnrollmentID → Enrollment.EnrollmentID

EmergencyContacts.StudentID → Students.StudentID

效果與成果：

經過此階段正規化，已成功移除資料表中的**部分依賴與重複資訊**，並將每筆資料放置於最相關的實體表格中，達成 2NF。每張表皆以主鍵唯一標識，並透過外鍵連結保持完整的資料結構。

---

### c. 正規化過程說明：從 2NF 到 3NF

在完成 2NF 後，我們已經消除了主鍵的部分依賴（partial dependency），每張表中的非主鍵欄位都完全依賴於主鍵。然而，仍可能存在傳遞依賴（transitive dependency），也就是某些非主鍵欄位依賴於其他非主鍵欄位，進而間接依賴主鍵。

#### 傳遞依賴問題：

##### 學生表中的 **ClassID → ClassName, Tuition**

如果 Students 表同時存放 ClassID 和 ClassName，那 ClassName 就是依賴 ClassID 而不是 StudentID。

處理方式：將班級資訊獨立成 Classes 表，只在 Students 表保留 ClassID 作為外鍵。

##### 老師與班級關係表中 **ClassID → Schedule**

若 TeacherClass（或 Teaches）表中同時記錄班級時段（Schedule），但時段其實是固定對應 ClassID，這就是傳遞依賴。

處理方式：將 Schedule 留在 Classes 表中，從授課關聯表中移除。

##### 付款摘要表中 **StudentID → EnglishName, ClassName**

若在 Payment 表或 PaymentSummary 中放入學生英文名或班級名，也會造成傳遞依賴。

處理方式：資料應透過 JOIN 查詢即時取得，而非直接冗餘記錄在繳費表中。

## 最終資料庫設計：主鍵與外鍵一覽表

資料表名稱	主鍵 (Primary Key)	外鍵 (Foreign Key)
Students	StudentID	ClassID → Classes(ClassID)
Teachers	TeacherID	無
Classes	ClassID	無
Enrollment	EnrollID	StudentID → Students(StudentID) ClassID → Classes(ClassID)
Payment	PaymentID	EnrollID → Enrollment(EnrollID) StudentID → Students(StudentID)
PaymentLog	LogID	StudentID → Students(StudentID)
PaymentSummary (*)	(可用 StudentID + PayDate)	無 (為報表用 View，無實體外鍵)
Teaches	TeacherID + ClassID	TeacherID → Teachers(TeacherID) ClassID → Classes(ClassID)
WorkHours	WorkID (或 TeacherID + WorkDate)	TeacherID → Teachers(TeacherID)
EmergencyContacts	ContactID	StudentID → Students(StudentID)
TeacherClass (可省略)	TeacherID + ClassID	TeacherID、ClassID (與 Teaches 重複)



## 2. 資料表建立與基本操作

### a. Schema 建立與初始資料插入 (CREATE TABLE, INSERT)。

<pre>●CREATE TABLE Classes (     ClassID    VARCHAR(10) PRIMARY KEY,     ClassName  VARCHAR(50) NOT NULL,     Tuition    DECIMAL(8,2),     Schedule   VARCHAR(50) );</pre>	CREATE TABLE CLASSES
<pre>●INSERT INTO Classes (ClassID, ClassName, Tuition, Schedule) ('C01', 'Puppy', 9000.00, '每週二、五 16:30-18:30'), ('C02', 'Sun', 9000.00, '每週二、五 16:30-18:30'), ('C03', 'Grass', 10800.00, '每週一、四 16:30-18:30'), ('C04', 'Nike', 14400.00, '每週六 09:00-12:00'), ('C05', 'Capybara', 10800.00, '每週一、四 19:00-21:00');</pre>	INSERT INTO CLASSES
<pre>●CREATE TABLE Students (     StudentID   VARCHAR(10) PRIMARY KEY,     ChineseName VARCHAR(50) NOT NULL,     EnglishName VARCHAR(50) NOT NULL,     Gender       CHAR(1) CHECK (Gender IN ('M', 'F')),     BirthDate    DATE NOT NULL,     Address      VARCHAR(100),     ClassID      VARCHAR(10),     FOREIGN KEY (ClassID) REFERENCES Classes(ClassID) );</pre>	CREATE TABLE STUDENTS
<pre>●INSERT INTO Students (StudentID, ChineseName, EnglishName, Gender, BirthDate, Address, ClassID) ('S001', '吳品露', 'Nova', 'F', '2017-01-23', '新北市蘆洲區中正路123號5樓', 'C01'), ('S002', '蘇宜羽', 'Ivy', 'F', '2017-07-18', '新北市蘆洲區光華路71號8樓', 'C01'), ('S003', '侯育均', 'Yoda', 'F', '2015-05-09', '新北市蘆洲區光復路59號2樓', 'C02'), ('S004', '蔡浩宇', 'Alston', 'M', '2017-03-10', '新北市蘆洲區中正路310號3樓', 'C02'), ('S005', '林廷毅', 'Ryan', 'M', '2015-08-26', '新北市蘆洲區中山二路20號5樓', 'C02'), ('S006', '林廷安', 'Iing', 'F', '2013-11-05', '新北市蘆洲區中山二路20號5樓', 'C03'), ('S007', '江亮穎', 'Bella', 'F', '2013-10-29', '新北市蘆洲區中原路102號9樓', 'C03'), ('S008', '黃晟齊', 'Nicky', 'M', '2011-05-01', '新北市蘆洲區三民路100號5樓之1', 'C04'), ('S009', '黃育瑄', 'Amy', 'F', '2010-12-31', '新北市蘆洲區中正路132號1樓', 'C04'), ('S010', '姜郁祺', 'Linda', 'F', '2015-08-22', '新北市蘆洲區長榮路82號2樓', 'C05');</pre>	INSERT INTO CLASSES

(以下略)

### b.索引與效能考量：為常用查詢欄位建立索引。

English \* <localhost> Console x Students EmergencyContacts

```
CREATE INDEX idx_student_name ON Student(Name);
CREATE INDEX idx_class_teacher ON Class(TeacherID);
CREATE INDEX idx_enroll_student ON Enrollment(StudentID);
CREATE INDEX idx_enroll_class ON Enrollment(ClassID);
```

c. 交易 (Transaction) 機制：BEGIN、COMMIT、ROLLBACK 範例。

BEGIN、COMMIT、ROLLBACK 範例	
前端程式碼	
<div><div>sJS app.jsJS Enrollment.jsJS Payment.jsJS enroll.jsXJS server.js{}</div><div>routes &gt; JS enroll.js &gt; ... 1 const express = require('express'); 2 const router = express.Router(); 3 const sequelize = require('../database'); 4 const Enrollment = require('../models/Enrollment'); 5 const Payment = require('../models/Payment'); 6 7 router.post('/', async (req, res) =&gt; { 8   const { studentId, classId, term, amount, method } = req.body; 9 10  const t = await sequelize.transaction(); 11  try { 12    const enroll = await Enrollment.create({ 13      StudentID: studentId, 14      ClassID: classId, 15      Term: term, 16      EnrollDate: new Date() 17    }, { transaction: t }); 18 19    await Payment.create({ 20      EnrollID: enroll.EnrollID, 21      Amount: amount, 22      PayDate: new Date(), 23      Method: method 24    }, { transaction: t }); 25 26    await t.commit(); 27    res.json({ message: '報名與繳費成功!' }); 28  } catch (err) { 29    await t.rollback(); 30    res.status(500).json({ message: '交易失敗，已回滾。', error: err.message }); 31  } 32  }); 33 34  module.exports = router;</div></div>	
學生報名 與繳費 初始畫面	<div><div>學生報名與繳費</div><div><div>StudentID:</div><div>ClassID:</div><div>Term: 2025春季</div><div>繳費金額:</div><div>繳費方式: 現金</div><div>送出報名與繳費</div></div></div>

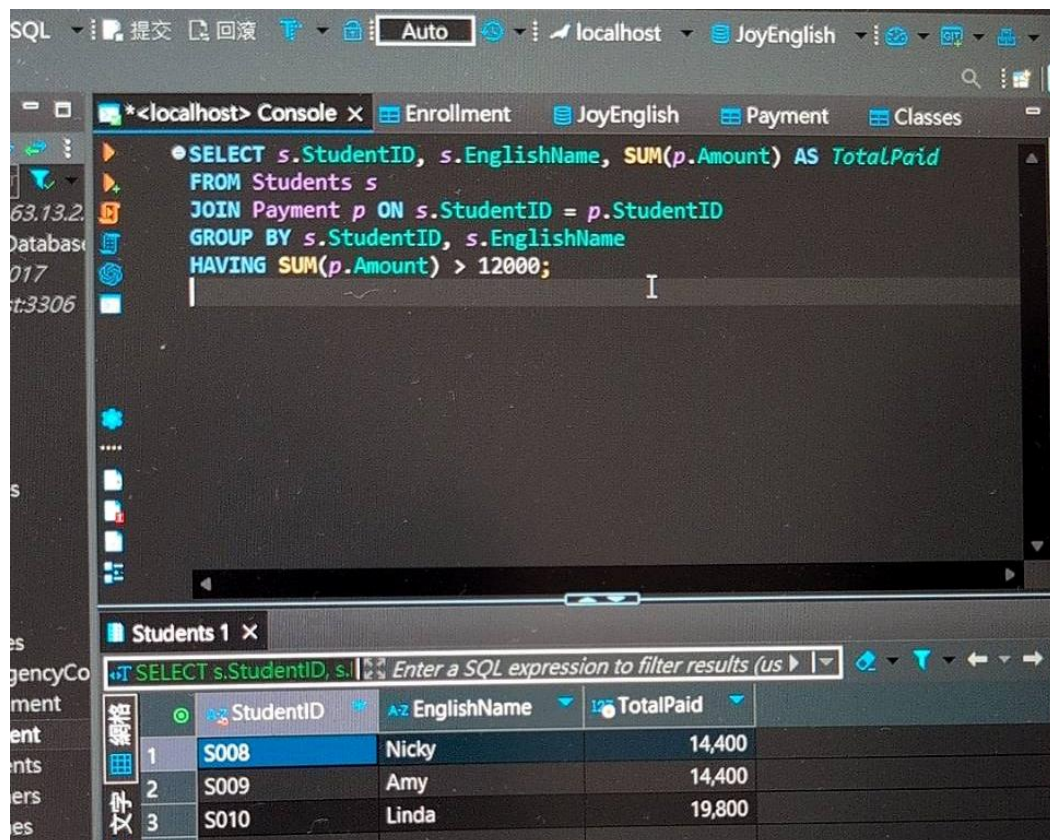
### 3. 進階 SQL 功能應用

a. 複雜查詢與子查詢：聚合函數、GROUP BY、HAVING、視圖 (View)。

#### 聚合函數

```
CREATE PROCEDURE JoyEnglish.CalcSalary(IN tid VARCHAR(10))
BEGIN
  SELECT
    t.TeacherID,
    t.EnglishName,
    SUM(w.Hours * t.HourlyRate) AS MonthlySalary
  FROM
    Teachers t
    JOIN WorkHours w ON t.TeacherID = w.TeacherID
  WHERE
    t.TeacherID = tid
  GROUP BY
    t.TeacherID, t.EnglishName;
END;
```

#### 顯示繳費超過 12000 的學生



The screenshot shows a SQL IDE interface. The top toolbar includes buttons for '提交' (Submit), '回滾' (Rollback), and 'Auto'. The database connection is set to 'localhost' and the schema is 'JoyEnglish'. The query editor contains the following SQL statement:

```
SELECT s.StudentID, s.EnglishName, SUM(p.Amount) AS TotalPaid
FROM Students s
JOIN Payment p ON s.StudentID = p.StudentID
GROUP BY s.StudentID, s.EnglishName
HAVING SUM(p.Amount) > 12000;
```

The results are displayed in a table titled 'Students 1'. The table has three columns: 'StudentID', 'EnglishName', and 'TotalPaid'. The results are as follows:

	StudentID	EnglishName	TotalPaid
1	S008	Nicky	14,400
2	S009	Amy	14,400
3	S010	Linda	19,800



## 建立視圖

```
Auto | localhost | JoyEnglish | *<localhost> Console x | Enrollment | JoyEnglish | Payment | Classes  
CREATE VIEW PaymentSummary AS  
SELECT s.StudentID, s.EnglishName, c.ClassName, p.Amount, p.PayDate  
FROM Students s  
JOIN Payment p ON s.StudentID = p.StudentID  
JOIN Enrollment e ON s.StudentID = e.StudentID  
JOIN Classes c ON e.ClassID = c.ClassID;
```

```
localhost | JoyEnglish | *<localhost> Console x | Enrollment | JoyEnglish | Payment | Classes  
SELECT * FROM PaymentSummary WHERE StudentID = 'S003';
```

PaymentSummary 1 x

SELECT \* FROM PaymentSummary WHERE StudentID = 'S003';

	A-Z StudentID	A-Z EnglishName	A-Z ClassName	123 Amount	PayDate
1	S003	Yoda	Sun	9,000	2025-03-06

b. Stored Procedure / Function：實作商業邏輯或報表計算。

## Stored Procedure

```
JoyEnglish | *<localhost... x | Students | EmergencyCo... | P  
CALL CalcSalary('T001');
```

Teachers 1 x | Statistics 1

CALL CalcSalary('T001');

	A-Z TeacherID	A-Z EnglishName	123 MonthlySalary
1	T001	Stephanie	3,600

c.Trigger：自動處理插入、更新、刪除事件。

Trigger

JoyEnglish \*<localhost... PaymentSummary Payment CalcSalary \*<localhost... 6

```
CREATE TRIGGER trg_PaymentLog_AfterInsert
AFTER INSERT ON Payment
FOR EACH ROW
INSERT INTO PaymentLog (StudentID, LogMessage, CreatedAt)
VALUES (
    NEW.StudentID,
    CONCAT('完成繳費，金額：', NEW.Amount, '，方式：', NEW.Method),
    NOW()
);
```

JoyEnglish \*<localhost... Payment CalcSalary \*<localhost... Payment.trg...

```
INSERT INTO Payment (EnrollID, StudentID, PayDate, Amount, Method)
VALUES (1, 'S001', '2025-06-18', 9000, '轉帳');
```

Auto localhost JoyEnglish \*<localhost... Payment.trg... 7

```
SELECT * FROM PaymentLog WHERE StudentID = 'S001';
```

PaymentLog 1

SELECT \* FROM PaymentLog

LogID	StudentID	LogMessage	CreatedAt
1	S001	完成繳費，金額：9000.00，方式：轉帳	2025-06-22 10:44:39.000

d.安全性與權限管理 (可選)：建立 ROLE、分配權限。

本系統為強化資料存取安全性，針對不同角色設計權限範圍。

例如建立 **accounting** 角色，僅授予其對繳費資料之查詢與新增權限，避免誤刪或非法修改資料，提高資料一致性與安全性。

建立角色→授權繳費功能→建立一個使用者並分配角色→檢查角色與權限

```
CREATE ROLE 'accounting';
```

```
GRANT SELECT, INSERT ON JoyEnglish.Payment TO 'accounting';
```

```
CREATE USER 'joyuser'@'localhost' IDENTIFIED BY 'joypass';
```

```
GRANT 'accounting' TO 'joyuser'@'localhost';
```

```
SHOW GRANTS FOR 'joyuser'@'localhost';
```

Results 1 ×

SHOW GRANTS FOR 'joyuser'@'localhost'

	Grants for joyuser@localhost
1	GRANT `accounting` TO `joyuser`@`localhost`
2	GRANT USAGE ON *.* TO `joyuser`@`localhost` IDENTIFIED BY PASSWORD '*88'

#### 4. 功能測試

a.功能查詢清單：設計並附上 SQL 與執行結果。

查詢 Sun 班所有學生

```
*-<localhost> Console x
SELECT s.StudentID, s.EnglishName
FROM Students s
JOIN Classes c ON s.ClassID = c.ClassID
WHERE c.ClassName = 'Sun';
```

Students 1 x

SELECT s.StudentID, s.EnglishName FROM

	StudentID	EnglishName
1	S003	Yoda
2	S004	Alston
3	S005	Ryan

查詢某位學生的上課與繳費歷史

```
*-<localhost> Console x
SELECT s.StudentID, c.ClassName, SUM(p.Amount) AS TotalPaid, MAX(p.PayDate) AS LastPaymentDate
FROM Students s
JOIN Enrollment e ON s.StudentID = e.StudentID
JOIN Classes c ON e.ClassID = c.ClassID
LEFT JOIN Payment p ON e.EnrollID = p.EnrollID
WHERE s.StudentID = 'S003'
GROUP BY s.StudentID, c.ClassName;
```

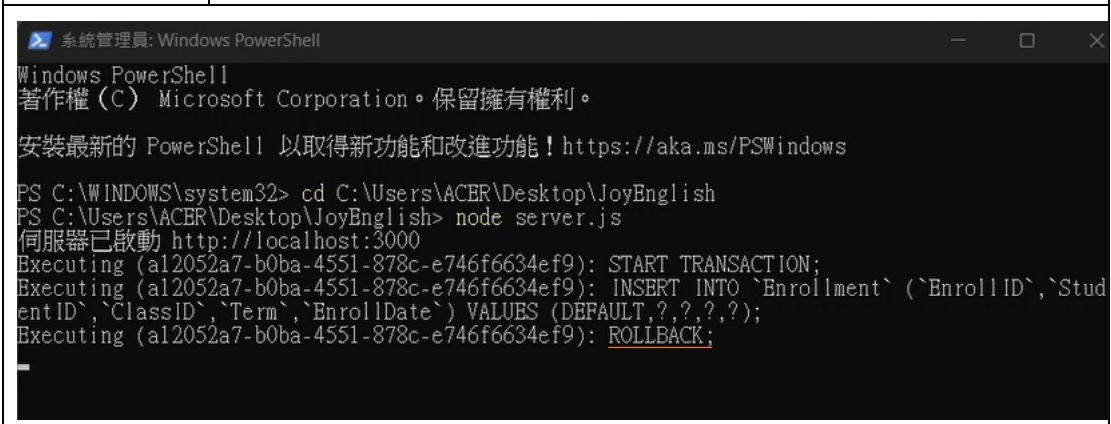

Students(+) 1 x

SELECT s.StudentID, c.ClassName, SUM

	StudentID	ClassName	TotalPaid	LastPaymentDate
1	S003	Sun	9,000	2025-03-06



b.資料一致性與異常情況測試：外鍵約束、Transaction Rollback。

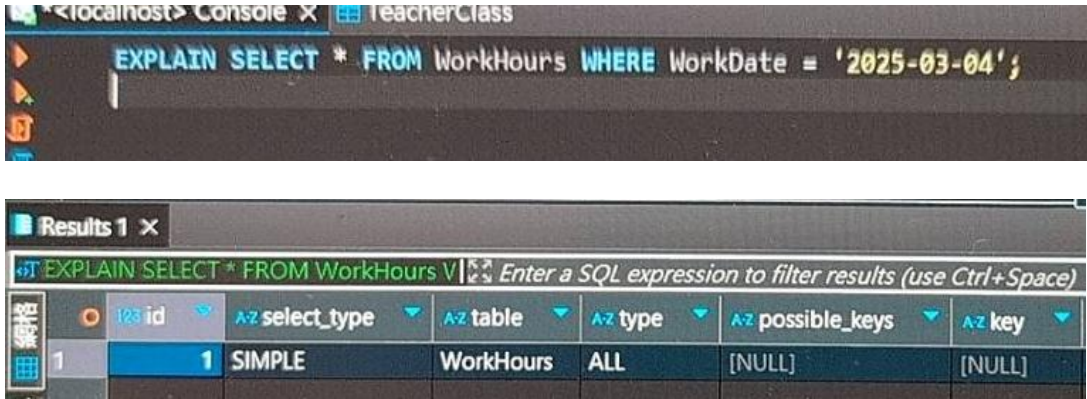
<p>學生報名 與繳費 交易失敗 回滾畫面</p>	<div><h3>學生報名與繳費</h3><div><div>StudentID:</div><div></div><div>ClassID:</div><div>C01</div><div>Term:</div><div>2025春季</div><div>繳費金額:</div><div>9000</div><div>繳費方式:</div><div>現金</div><div>送出報名與繳費</div></div><p>交易失敗，已回滾。</p></div>
	<div><pre>系統管理員: Windows PowerShell Windows PowerShell 著作權 (C) Microsoft Corporation。保留擁有權利。 安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows PS C:\WINDOWS\system32&gt; cd C:\Users\ACER\Desktop\JoyEnglish PS C:\Users\ACER\Desktop\JoyEnglish&gt; node server.js 伺服器已啟動 http://localhost:3000 Executing (a12052a7-b0ba-4551-878c-e746f6634ef9): START TRANSACTION; Executing (a12052a7-b0ba-4551-878c-e746f6634ef9): INSERT INTO `Enrollment` (`EnrollID`,`StudentID`,`ClassID`,`Term`,`EnrollDate`) VALUES (DEFAULT,?,?,?,?); Executing (a12052a7-b0ba-4551-878c-e746f6634ef9): ROLLBACK;</pre></div>
<p>學生報名 與繳費 交易成功 執行畫面</p>	<div><h3>學生報名與繳費</h3><div><div>StudentID:</div><div>S010</div><div>ClassID:</div><div>C01</div><div>Term:</div><div>2025春季</div><div>繳費金額:</div><div>9000</div><div>繳費方式:</div><div>現金</div><div>送出報名與繳費</div></div><p>報名與繳費成功！</p></div>
	<div><pre>系統管理員: Windows PowerShell Windows PowerShell 著作權 (C) Microsoft Corporation。保留擁有權利。 安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows PS C:\WINDOWS\system32&gt; cd C:\Users\ACER\Desktop\JoyEnglish PS C:\Users\ACER\Desktop\JoyEnglish&gt; node server.js 伺服器已啟動 http://localhost:3000 Executing (9c62b587-2026-4487-8135-e39d70f3d9f7): START TRANSACTION; Executing (9c62b587-2026-4487-8135-e39d70f3d9f7): INSERT INTO `Enrollment` (`EnrollID`,`StudentID`,`ClassID`,`Term`,`EnrollDate`) VALUES (DEFAULT,?,?,?,?); Executing (9c62b587-2026-4487-8135-e39d70f3d9f7): INSERT INTO `Payment` (`PaymentID`,`EnrollID`,`Amount`,`PayDate`,`Method`) VALUES (DEFAULT,?,?,?,?); Executing (9c62b587-2026-4487-8135-e39d70f3d9f7): COMMIT;</pre></div>



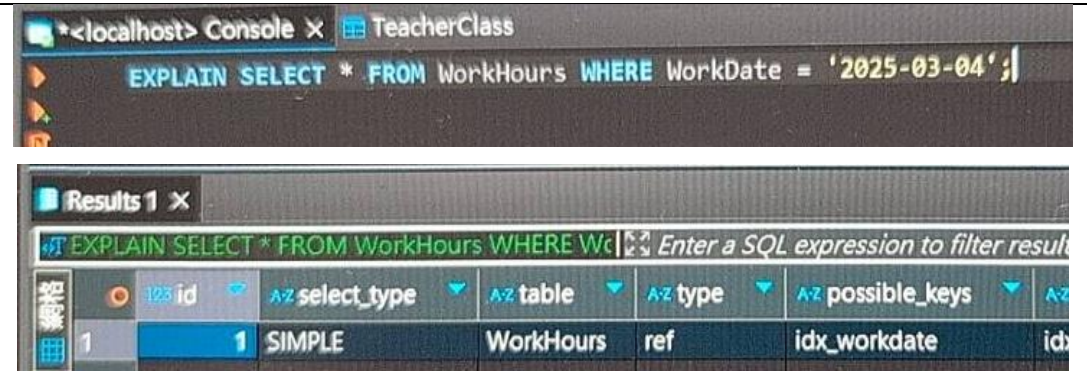
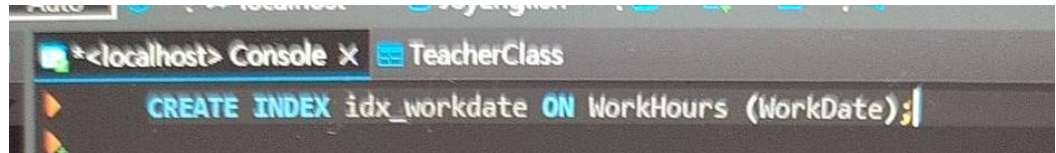
c.效能測試 (可簡化)：比較索引前後查詢時間。

此系統對 WorkHours.WorkDate 建立索引。原查詢執行計畫為 type=ALL (全表掃描)；建立索引後為 type=ref，表示已使用索引查找，效能提升明顯。

加入索引前 type=ALL→加入索引→加入索引後 type=ref



	id	select_type	table	type	possible_keys	key
1	1	SIMPLE	WorkHours	ALL	[NULL]	[NULL]



	id	select_type	table	type	possible_keys	key
1	1	SIMPLE	WorkHours	ref	idx_workdate	idx_workdate

##### 5.組員分工說明:

410630460 徐蕙蓁	資料庫設計、SQL 實作、DBeaver 操作、文件撰寫與報告
---------------	---------------------------------