國立虎尾科技大學資訊工程系

**資料庫期末報告**

參與成員： 宋協燦 高浩城

張承翰 郭建杰

指導教授簽名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中華民國 年 月 日

目 錄

[一、 摘要 4](#_Toc199262202)

[二、 應用情境 5](#_Toc199262203)

[三、 系統需求 6](#_Toc199262204)

[四、 使用案例 7](#_Toc199262205)

[五、 資料庫建立 8](#_Toc199262206)

[5.1 users 使用者資料表 8](#_Toc199262207)

[5.2 categories 交易分類表 9](#_Toc199262208)

[5.3 transactions 交易紀錄表 11](#_Toc199262209)

[5.4 budgets 每月預算表 13](#_Toc199262210)

[5.5 saving\_goals 儲蓄目標表 15](#_Toc199262211)

[六、 資料庫查詢方法 17](#_Toc199262212)

[七、 VIEW表建立 25](#_Toc199262213)

[八、 分工 30](#_Toc199262214)

圖目錄

[圖 1：Users資料表實際建置 9](#_Toc199262006)

[圖 2：categories資料表實際建置 10](#_Toc199262007)

[圖 3：transactions資料表實際建置 12](#_Toc199262008)

[圖 4：budgets資料表實際建置 14](#_Toc199262009)

[圖 5：saving\_goals資料表實際建置 16](#_Toc199262010)

[圖 6：使用者註冊畫面 17](#_Toc199262011)

[圖 7：使用者登入畫面 18](#_Toc199262012)

[圖 8：新增類別畫面 19](#_Toc199262013)

[圖 9：新增交易畫面 20](#_Toc199262014)

[圖 10：每月預算表畫面 22](#_Toc199262015)

[圖 11：新增儲蓄目標畫面 23](#_Toc199262016)

[圖 12：saving\_goal\_status SQL 25](#_Toc199262017)

[圖 13：user\_transaction\_history SQL 26](#_Toc199262018)

[圖 14：user\_financial\_sunnary SQL 27](#_Toc199262019)

[圖 15：user\_category\_summary SQL 28](#_Toc199262020)

[圖 16：財務顯示畫面 29](#_Toc199262021)

**國立虎尾科技大學資訊工程系(科)專題報告**

**題目：**

指導老師：

專題參與人員：

班級：

1. 摘要

此專題利用Amazon RDS建立一個雲端資料庫，可以供不同使用者註冊帳號享有自己的雲端記帳系統，可以記錄自己的總資產並會隨著新增的收入支出去改變資產數，也可以對自己的交易去新增不同的類別，像是飲食、交通、公司開銷、旅遊等可以依照使用者需求去自訂。紀錄交易時系統會依照使用者選擇的是收入或支出去做不同的行為，例如這筆交易是收入，就將此交易的金額加入總資產，讓總資產可以隨著交易的紀錄靈活變動。用戶還能建立自己每個月的預算金額是多少，例如在2025年的1月在飲食上的預算是多少錢，系統也會記錄目前已經花了多少錢。除了上述功能，還能新增儲蓄目標讓使用者自行設定不同目標，像是筆電需要多少錢、從什麼時候開始、何時結束，系統會顯示目前已經儲蓄的金額跟該目標是否已經完成。

1. 應用情境

使用對象可以是學生、上班族、任何想要紀錄並管理自己財務的使用者都可以使用此系統。

1. 系統需求
2. 使用案例
3. 資料庫建立

確定好各項系統需求跟應用情境後，根據上述來建立所需要的資料庫，該專題的資料庫總共建立以下users、categories、transactions、budgets、saving\_goals五種，透過資料表間的關係達成系統所需要的功能，以下是資料表的細節。

* 1. users 使用者資料表

表 1：user資料表

|  |  |  |  |
| --- | --- | --- | --- |
| **欄位名稱** | **資料型別** | **限制條件** | **說明** |
| user\_id | INTEGER | PRIMARY KEY | 使用者 ID |
| user\_account | CHAR(50) | UNIQUE, NOT NULL | 使用者帳號 |
| name | CHAR(50) | NOT NULL | 姓名 |
| total\_assets | INTEGER | DEFAULT 0 CHECK (total\_assets >= 0) | 總資產 |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | 建立時間 |

user\_id是負責辨識目前是哪個使用者使用記帳系統的，註冊時只要成功資料表就會自動生成ID，系統主要透過這個ID去做當前需要資料的查詢，像是使用者現在要查看自己的交易紀錄，系統就會利用該ID從資料庫查詢該ID的交易紀錄來給用戶查看，確保不會查詢到不同於該ID的資料。為了避免重複所以設置主鍵，確保資料表內的資料不會有兩個同樣ID的使用者。

user\_account是儲存使用者帳號的，讓使用者可以擁有自己的記帳空間，用帳戶功能保護各個使用者間的隱私，確保不同人不會洩漏資訊，該資料表只儲存帳號需要調用密碼資料庫來做對應，才能讓用戶順利登入。每個帳號只能註冊一次並且不能沒有帳號的輸入。

name是讓使用者可以自行取符號喜好的名字，只要符合規則資料表就能允許該名稱的建立。因為系統還是需要有個名稱可以顯示，所以名稱不能為空。

total\_assets是使用者的總資產，會根據交易的收入支出更改，一開始是以0當作預設值。

created\_at是這位使用者建立的時間是什麼時候，可以讓管理員或使用者自己查看帳號是什麼時候建立的。預設是建立當下的日期與時間。

表 2：user資料表的完整性限制

|  |  |
| --- | --- |
| **欄位名稱** | **完整性限制條件** |
| user\_id | 系統依照使用者建立順序給予整數ID，從1開始，每多一個使用者就+1 |
| user\_account | 格式為<local-part>@<domin>。<local-part>的限制條件只能包含英文字母Aa~Zz、數字0~9、特殊符號!#$%&’\*+-/=?^\_{|}~.`，點號不能作為開頭或結尾、不可連續出現，系統不接受” “。<domin>結構以點號進行分割，每段長度小於等於63字元，全部不超過255字元，只能包含英文字母Aa~Zz、數字0~9、特殊符號-，但特殊符號-不能當作開頭、結尾，系統不接受(test)註釋的方式。 |
| name | 只能含英文字母Aa~Zz，不可含中文、特殊符號，長度最多50 |
| total\_assets | 只能是數字0~9組成，不能為負數或其他特殊符號，至少為0 |
| created\_at | 格式為 yyyy- mm-dd HH:MM:SS，系統會自動帶入建立當下時間 |

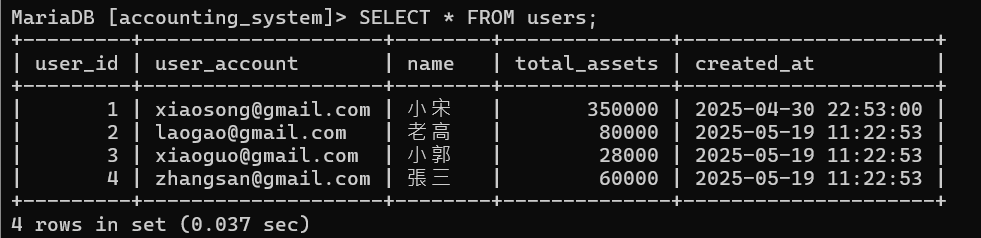


圖 1：Users資料表實際建置

* 1. categories 交易分類表

表 3： categories資料表

|  |  |  |  |
| --- | --- | --- | --- |
| **欄位名稱** | **資料型別** | **限制條件** | **說明** |
| category\_id | INTEGER | PRIMARY KEY | 分類 ID |
| user\_id | INTEGER | FOREIGN KEY → users(user\_id) | 使用者 ID |
| name | CHAR(50) | NOT NULL, UNIQUE(user\_id, name) | 分類名稱 |

category\_id是使用者建立分類ID時會自動編號的ID，作用是系統方便統計目前資料庫中總共被建立了多少類別。因為不能重複，所以建立為主鍵。

user\_id是users資料表裡的資料，為了辨識當前建立分類的是哪個使用者，以防不同使用者之間可以亂建立類別。因為是參考其他資料表的資料，所以設定為外鍵。

name是使用者設定的分類是叫什麼名稱，例如交通、醫療、飲食、房租等不同符合規則的分類。這個欄位不能是空的，必須要有此分類的名稱，而同個使用者不能建立一模一樣的分類名稱，避免系統出現錯誤，像是使用者1號不能建立兩個飲食的分類，系統會根據user\_id判斷目前是否已經建立過飲食，如果已經存在就會告訴使用者1號錯誤。

表 4：category完整性限制

|  |  |
| --- | --- |
| **欄位名稱** | **完整性限制條件** |
| category\_id | 系統會根據使用者建立的類別去照順序編號，編號是從1開始的整數，只要分類建立就會+1 |
| user\_id | 是users資料表中的ID，由數字1開始每次+1的編號 |
| name | 分類的名稱只能是中文字、英文字，不能有特殊符號與數字 |



圖 2：categories資料表實際建置

* 1. transactions 交易紀錄表

表 5：transaction資料表

|  |  |  |  |
| --- | --- | --- | --- |
| **欄位名稱** | **資料型別** | **限制條件** | **說明** |
| transaction\_id | INTEGER | PRIMARY KEY | 交易 ID |
| user\_id | INTEGER | FOREIGN KEY → users(user\_id) | 使用者 ID |
| category\_id | INTEGER | FOREIGN KEY → categories(category\_id) | 分類 ID |
| type | CHAR(7) | NOT NULL CHECK(type=‘Income’ OR type=‘Expense’) | 收入支出分類 |
| amount | INT | NOT NULL CHECK (amount >= 0) | 金額 |
| transaction\_date | DATE | NOT NULL | 交易日期 |
| description | CHAR(255) |  | 此項交易說明 |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | 建立時間 |

transaction\_id是使用者新增交易時最自動編號的ID，方便系統查看目前總交易數量跟出錯時可以透過編號查看是哪筆資料錯誤。因為不能重複，所以建立為主鍵。

user\_id是users資料表裡的資料，是系統用來辨識當前是哪個使用者建立該筆交易的，在查詢時也方便系統查到該使用者建立的交易。因為參考其他資料表的資料，設置為外鍵。

category\_id是category資料表裡的資料，是系統用來辨識當前的交易是屬於哪一種分類的，是飲食、交通等分類，系統會搭配user\_id來找到當前交易是使用者建立的哪一種分類。因為參考其他資料表的資料，設置為外鍵。

type是使用者在選擇是收入還是支出時決定，在存入資料庫前會依據使用者的選擇去存入不同的字串，方便檢查和查詢該筆資料是收入或支出。因為需要分辨兩種狀態，所以存入的字串只能是Income或Expense其中之一且不得為空。

amount就是該筆交易的金額為多少，方便使用者後續查看外，也讓系統可以對總資產進行數值的加減。該欄位不論收入支出都不得為負數，必須要大於0且不得為空。

transaction\_date是該筆交易是哪一天發生的日期，使用者就算錯過交易當天也可以自行設定交易日期確保記錄無誤。不能為空。

description使用者可以對這筆交易做說明，描述這筆是什麼東西，像是公司聚餐、高鐵票、五月薪資等，方便使用者查看紀錄時得知交易細節。可以為空，不強迫使用者一定要對交易做描述。

created\_at是新增這筆資料時的時間，可以讓使用者跟管理員查看這筆交易創建的時間。預設是新增資料當下日期與時間。

表 6：transactions完整性限制

|  |  |
| --- | --- |
| **欄位名稱** | **完整性限制條件** |
| transaction\_id | 系統會根據每一筆交易建立的順序去給該交易訂單一個編號，該編號是一個整數，從1開始的，每有一筆新訂單就+1 |
| user\_id | 由整數組成，不包含特殊符號、文字 |
| category\_id | 由整數組成，不包含特殊符號、文字 |
| type | 只能是Income或Expense兩種英文單字，不能含有數字、特殊符號、除這兩個英文單字外的英文字母 |
| amount | 金額只能由0到9的數字去組成，不能為負數必須大於等於0，也不能包含文字、特殊符號 |
| transaction\_date | 格式為 yyyy-mm-dd 。yyyy年是由0~9數字組成，第一位不得為0、mm月如果為個位數月份第一位必須輸入0且是由1~12數字組成、dd日如果為個位數日第一位必須輸入0且是由1~31數字組成，如果當月沒有31號，在存入資料庫前，系統會自動更改為30號。 |
| description | 說明可以由中文字、英文字Aa到Zz、數字0到9組成，但不能包含特殊符號，長度最多為255 |
| created\_at | 格式為 yyyy-mm-dd ，系統會自動帶入建立當下時間 |

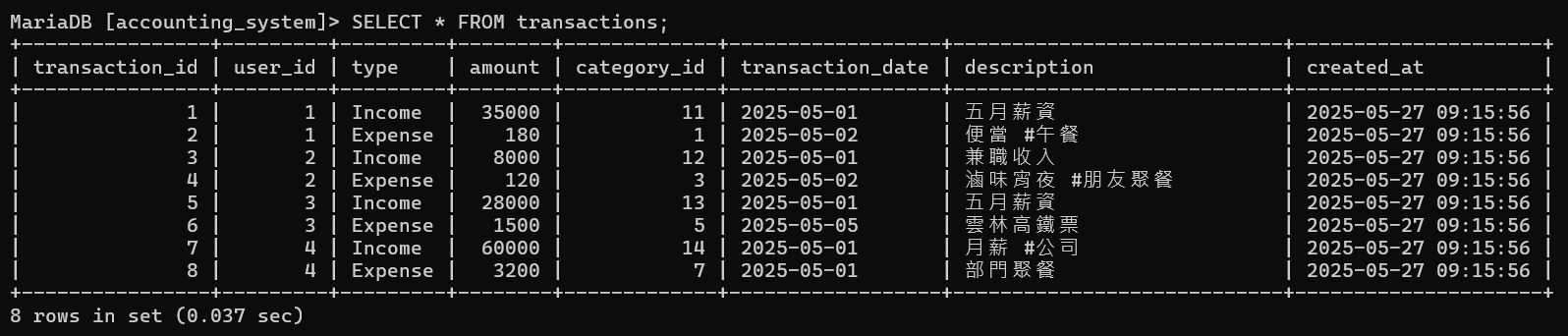


圖 3：transactions資料表實際建置

* 1. budgets 每月預算表

表 7：budget資料表

|  |  |  |  |
| --- | --- | --- | --- |
| **欄位名稱** | **資料型別** | **限制條件** | **說明** |
| budget\_id | INTEGER | PRIMARY KEY | 預算 ID |
| user\_id | INTEGER | FOREIGN KEY → users(user\_id) | 使用者 ID |
| category\_id | INTEGER | FOREIGN KEY → categories(category\_id) | 分類 ID |
| year | INTEGER | NOT NULL | 年份 |
| month | INTEGER | NOT NULL, CHECK (month BETWEEN 1 AND 12) | 月份 |
| budget\_limit | INTEGER | NOT NULL, CHECK (budget\_limit >= 0) | 分類預算金額 |
| spent\_amount | INTEGER | DEFAULT 0 CHECK (spent\_amount >= 0) | 已花費預算 |

budget是使用者建立預算後系統會自動編碼的ID，都是由整數組成從1開始，只要建立一筆就會+1。因為不能重複所以設置主鍵。

user\_id是users資料表裡的資料，是系統用來辨識當前是哪個使用者建立這筆預算表的，在查詢時也方便查找當前用戶的預算表。因為參考其他資料表的資料，設置為外鍵。

category\_id是category資料表裡的資料，是系統用來辨識當前的預算是屬於哪一種分類的，是飲食、交通等分類，系統會搭配user\_id來找到該用戶建立的類別。因為參考其他資料表的資料，設置為外鍵。

year讓使用者輸入哪一年要建立的預算表，可以設定未來的年份提早做準備。不能為空。

Month讓使用者輸入是哪一個月要建立預算表，搭配年份讓使用者清楚的知道這筆預算表是幾年幾月所需要的預算表。不能為空且只能有數字1~12所組成。

budget\_limit是該預算表中的預算金額為多少，讓使用者自訂想要分配多少金額在該年該月。不能為空且確保欄位數值大於等於0。

spent\_amount是使用者目前已經在這個月花多少預算了，清楚讓用戶知道這個月已經花了多少錢。預設值是0，也要確保該欄位是大於等於0。

表 8：budget完整性限制

|  |  |
| --- | --- |
| **欄位名稱** | **完整性限制條件** |
| budget\_id | 系統會根據每個使用者建立預算表照順序給該表編號，此編號為由1開始的整數，有新的預算表就+1 |
| user\_id | 由整數組成，不包含特殊符號、文字 |
| category\_id | 由整數組成，不包含特殊符號、文字 |
| year | 年份為西元年，年份由四個數字組成，開頭不得為0，後三碼可以是0~9的數字去組成，不得含有文字、特殊符號 |
| month | 只能由數字1到12任一，不能含有文字、特殊符號 |
| budget\_limit | 只能有數字0到9去組成，不能含有文字、特殊符號，且該數字必須大於等於0，不得為負數 |
| spent\_amount | 只能有數字0到9去組成，不能含有文字、特殊符號，且該數字必須大於等於0，不得為負數 |

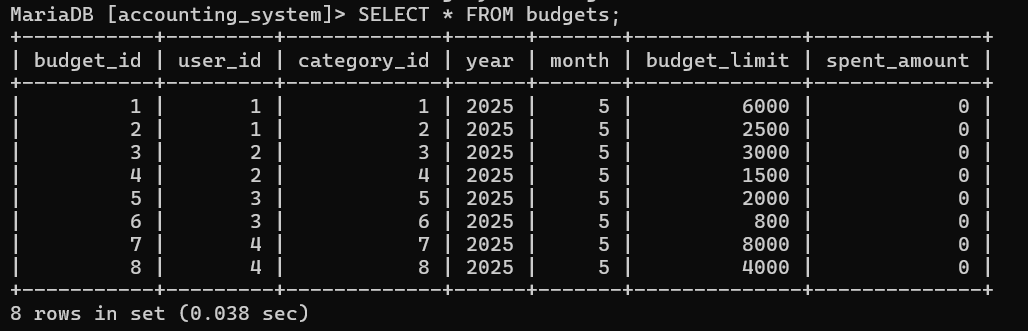


圖 4：budgets資料表實際建置

* 1. saving\_goals 儲蓄目標表

表 9：saving\_goals資料表

|  |  |  |  |
| --- | --- | --- | --- |
| **欄位名稱** | **資料型別** | **限制條件** | **說明** |
| goal\_id | INTEGER | PRIMARY KEY | 目標 ID |
| user\_id | INTEGER | FOREIGN KEY → users(user\_id) | 使用者 ID |
| name | CHAR(50) | NOT NULL | 儲蓄目標名稱 |
| target\_amount | INTEGER | NOT NULL, CHECK (target\_amount > 0) | 目標金額 |
| current\_amount | INTEGER | DEFAULT 0, CHECK (current\_amount >= 0) | 目前已儲蓄金額 |
| start\_date | DATE | NOT NULL | 儲蓄開始日期 |
| end\_date | DATE | NOT NULL | 儲蓄結束日期 |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP | 建立時間 |
| status | CHAR(20) | DEFAULT ‘Active’ CHECK (status IN (‘Active’, ‘Completed’)) | 目標完成狀態 |

goal\_id只要使用者建立新的儲蓄目標系統就會自動編號的ID，由整數組成，從1開始往上加總。因為不能重複，所以設定主鍵。

user\_id是users資料表裡的資料，是系統用來辨識當前是哪個使用者建立這儲蓄目標表，在查詢時也方便查找當前用戶的儲蓄目標表。因為參考其他資料表的資料，設置為外鍵。

name使用者可以按照規定取自己喜歡的目標名稱，方便使用者後續查看時，透過名稱簡單扼要的重點來知道這項儲蓄目標是為了什麼。不能為空，必須要可以分辨是什麼目標。

target\_amount是這項儲蓄目標的目標金額為多少，讓使用者自行設定要存到多少才算完成目標。不能為空且必須大於0。

current\_amount是使用者當前已經儲蓄了多金額，可以讓系統判斷這項儲蓄目標是否已經達到目標。預設為0且該欄位必須大於等於0。

start\_date是這項儲蓄目標開始的日期，可以讓使用者按照自己的計劃去設定日期。不能為空。

end\_date是這項儲蓄目標結束的日期，可以讓使用者按照自己的計劃去設定日期。不能為空。

created\_at是這項儲蓄目標表建立的時間是什麼時候，可以讓管理員或使用者自己查看該表是什麼時候建立的。預設是建立當下的日期與時間。

status是如果該儲蓄目標完成了，就會設定為Completed，這樣使用者如果想查詢這項目標是否已經完成了，就可以直接利用該欄位判斷是否完成，如果還沒完成則會是Active。預設會是Active，而且該欄位只允許兩種英文單字Active、Completed。

表 10：saving\_goals完整性限制

|  |  |
| --- | --- |
| **欄位名稱** | **完整性限制條件** |
| goal\_id | 系統會根據使用者建立的每個目標去照順序編號，該編號是從1開始的整數，每有一個目標就+1 |
| user\_id | 由整數組成，不包含特殊符號、文字 |
| name | 目標名稱可以是英文、中文，不能含有數字、特殊符號，名稱長度最長可以到50個字 |
| target\_amount | 目標金額只能由0到9的數字組成，不能為負數，且不得包含文字與特殊符號 |
| current\_amount | 儲蓄金額只能由0到9的數字組成，不能為負數，且不得包含文字與特殊符號，預設為0 |
| start\_date | 格式為 yyyy-mm-dd 。yyyy年是由0~9數字組成，第一位不得為0、mm月如果為個位數月份第一位必須輸入0且是由1~12數字組成、dd日如果為個位數日第一位必須輸入0且是由1~31數字組成，如果當月沒有31號，在存入資料庫前，系統會自動更改為30號。 |
| end\_date | 格式為 yyyy-mm-dd 。yyyy年是由0~9數字組成，第一位不得為0、mm月如果為個位數月份第一位必須輸入0且是由1~12數字組成、dd日如果為個位數日第一位必須輸入0且是由1~31數字組成，如果當月沒有31號，在存入資料庫前，系統會自動更改為30號。 |
| created\_at | 建立時間會根據使用者建立該目標的當下去設定，格式為西元年yyy-mm-dd |
| status | 狀態是會根據該目標完成與否去給定，只能是英文字母的Active、Completed，不能含有數字、特殊符號、除這兩個單字外的英文字母 |

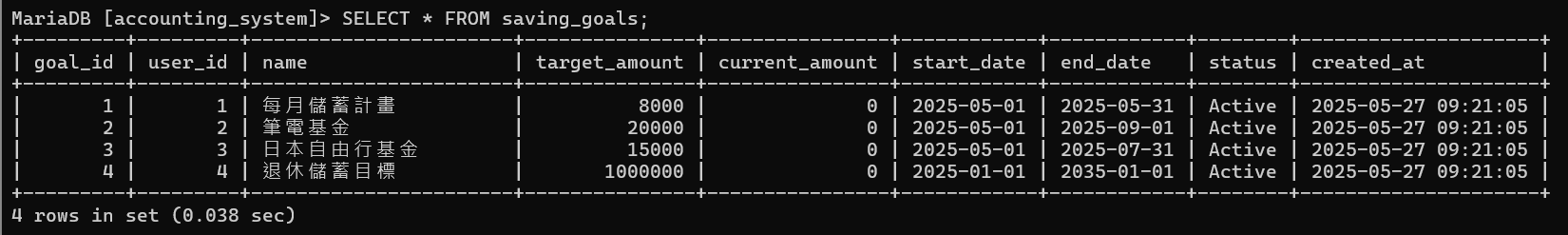


圖 5：saving\_goals資料表實際建置

1. 資料庫查詢方法

以下會解釋從登入註冊到顯示資料每一步會對資料庫做什麼新增、查詢。因為密碼是另外儲存，如果有提到密碼的會使用**password DB**來說明。舉例情境為小宋想要建立自己的帳戶，電子郵件為test@gmail.com，以下操作都預設為已經在五種資料表存在的資料庫裡的操作。

* **註冊畫面**

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 6：使用者註冊畫面

每個使用者在使用系統之前都必須要創建帳號，而創建帳號必須要先檢查資料庫裡是否有跟當前註冊帳號一樣的電子郵件，如果沒有一樣的就可以新增使用者、把密碼存在password DB。以下是在註冊畫面會對資料庫所做的操作與流程。

* **檢查帳號是否存在 ：**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com**’;

利用SELECT搭配WHERE的條件篩選去選出資料庫users資料表裡面的資料，抓取資料後就可以進一步去判斷說現在註冊的電子郵件是不是已經存在於資料庫了，如果已經存在了就回傳此電子郵件被註冊了，若沒有就可以進行下一步的新增。

* **建立新使用者：**

**INSERT INTO** users (user\_account, name,total\_assets) **VALUES (‘test@gmail.com’, ‘小宋’,30000)**

利用INSERT INTO將新的使用者插入資料庫中users的資料表，讓系統後續操作可以知道資料庫裡確定有這個使用者。

* **將密碼加密之後儲存至password DB：**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com**’;

**INSERT INTO** passwordDB.passwords (user\_id, password) **VALUES** (user\_id, ‘password’);

新增完使用者之後系統會自動編號user\_id，為了要能分辨密碼是哪個使用者的，需要再做一次查詢，查出**test@gmail.com**的user\_id是多少，最後在利用INSERT INTO 連同加密過後的密碼插入進password DB裡面。

* **登入畫面**

**一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。**

圖 7：使用者登入畫面

註冊完之後回到登入畫面讓使用者輸入帳號密碼，小宋輸入帳號**test@gmail.com、**密碼password，系統會先根據輸入的帳號查詢資料庫中users資料表裡是否有這位使用者，如果有就回傳對應的user\_id，反之就會回傳查無此用戶，有了回傳的user\_id就可以去password DB抓取對應的密碼來比對使用者輸入的密碼，一致就可以登入，錯誤則會擋住此次登入行為。

* **檢查使用者是否存在 SQL：**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com**’;

檢查是否在users資料表裡有帳號是test@gmail.com的user\_id，如果有就回傳user\_id，準備進行下一步，若沒有就回傳帳號未註冊。

* **檢查password DB裡的密碼**

**SELECT** password **FROM** password DB.passwords **WHERE** user\_id = ‘**1**’;

在password DB裡面抓取檢查帳號後回傳的user\_id的密碼，接下來的情況舉例都利用1示意為小宋帳號的user\_id，系統會對使用者輸入的密碼跟查詢到的密碼進行比對，如果一致就可以登入，反之則拒絕登入並顯示錯誤訊息。

* **新增類別**

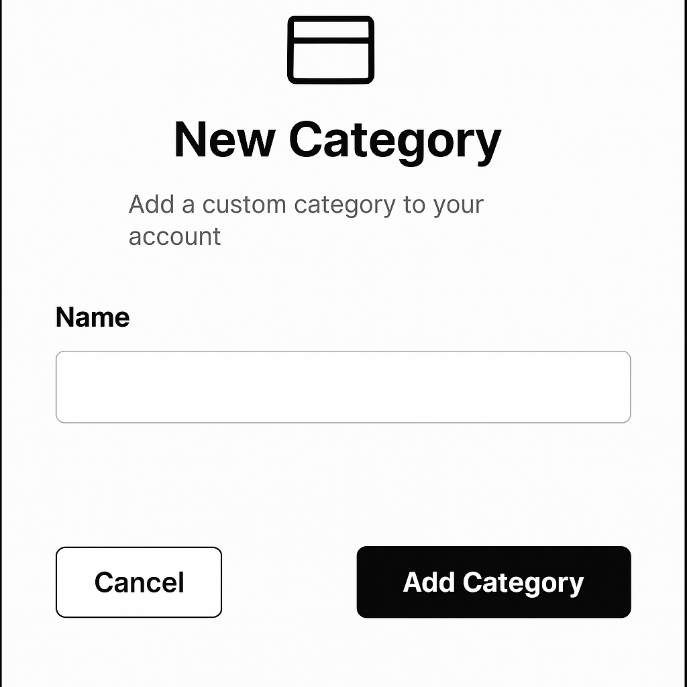


圖 8：新增類別畫面

使用者可以自己創建自己的類別，只要名稱符合規則就可以建立，讓使用者在新增交易時可以更好分類每一筆收入與支出是用於什麼地方，新增後系統會自動編號類別ID以利後續要抓取。

* **使用者查詢：**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com’**

系統新增類別前也需要知道當前使用者是誰，必須先查詢當前登入的電子郵件找出對應的使用者ID是多少。

* **新增類別**

**INSERT INTO** categories (user\_id, name) **VALUES**(1, ‘Food ‘)

根據查詢到的user\_id跟使用者輸入的名稱，確保符合只能有英文、中文長度限制為50後就可以新增到category資料表中，供後續使用者新增交易時可以選擇。

* **新增交易**

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 9：新增交易畫面

新增交易畫面可以先選擇這次交易是屬於Expense、Income，接著輸入這次交易的金額、日期跟可以為空的描述，還要選擇使用者自訂的類別後，就可以新增交易了。系統會根據當前是哪個使用者，選的收入與支出的分類與金額來新增transactions資料表的資料，並對users資料表中的總資產進行加減。

* **使用者查詢：**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com’;**

系統必須先知道要新增這筆交易的使用者ID是什麼，所以要先使用當前登入的帳號去做users的查詢，來得知當前使用者的ID是什麼才進行新增交易。

* **類別查詢：**

**SELECT** category\_id **FROM** categories **WHERE** user\_id = 1 **AND** name = ‘Food’;

系統需要知道類別ID，所以要去找categories資料表中當前使用者創建的類別ID是多少，使用剛剛查詢到的user\_id搭配使用者選擇的類別名稱去查詢category\_id是多少。

* **新增交易**

**INSERT INTO** transactions (user\_id, category\_id,type,amount,transaction\_date) **VALUES**(1,1,’Expense’,200,2025-05-27);

系統知道使用者的ID與類別ID後就可以新增交易資料了，根據使用者選擇的Expense、amount = 200、日期是2025-05-27、無描述，即可新增一筆交易資料，因為選擇的是支出所以會對users資料表中的total\_assets做減法。

* **更新總資產**

**UPDATE** users **SET** total\_assets = total\_assets - 200 **WHERE** user\_id = 1; **支出**

**UPDATE** users **SET** total\_assets = total\_assets + 200 **WHERE** user\_id = 1; **收入**

系統已經知道這筆交易的金額amount是200，所以利用UPDATE去更新users資料表中的總資產數量，資料庫會先讀取目前使用者的users裡的total\_assets的數值去做家減法。

* **新增每月預算表**

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 10：每月預算表畫面

使用者可以建立哪一年哪個月的每月預算表，提前分配好可以用的金額為多少，藉此讓自己可以更了解目前的財金狀況跟有效控管金錢流向。系統會根據當前使用者與其建立的類別來設定在哪一年哪一月的預算是多少。

* **查詢使用者**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com’;**

系統必須知道是哪個使用者新增的每月預算表，所以要用user\_account去查詢對應的使用者ID是什麼。

* **查詢類別**

**SELECT** category\_id **FROM** categories **WHERE** user\_id = 1 **AND** name = ‘Food’;

每月預算表是利用使用者建立的類別去做分類的，例如小宋在2025年5月分配飲食預算是3000元，所以除了user\_id還需要category\_id才能順利新增每月預算表。

* **新增每月預算表**

**INSERT INTO** budgets (user\_id, category\_id, year, month, budget\_limit, spent\_amount) **VALUES** (1,1,2025,5,5000,0);

有了user\_id、category\_id就可以依據使用者輸入的資料進行新增預算表，上述範例意思是使用者小宋在2025年5月的時候分配分類1的預算是5000，當前並沒有花費到預算，所以budget\_limit是5000表示預算上限是5000、spent\_amount是0表示還沒花到預算。

* **新增儲蓄目標**

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 11：新增儲蓄目標畫面

使用者可以根據新增畫面輸入目標名稱、開始日期、結束日期、目標金額、當前金額來新增儲蓄目標，系統會根據達成目標與否對使用者進行鼓勵。會利用當前使用者跟輸入的日期和金額對saving\_goal新增資料，並且會根據這項目標達成狀況來更改saving\_goal資料表中的status，預設是Active，如果達成了此項資料的status會變成Completed。

* **查詢使用者**

**SELECT** user\_id **FROM** users **WHERE** user\_account = ‘**test@gmail.com’;**

系統根據user\_account去尋找對應的使用者ID，這樣才能確定此項儲蓄目標是哪位使用者建立的，不造成混淆。

* **新增儲蓄目標**

**INSERT INTO** saving\_goal (user\_id, name, target\_amount, current\_amount, start\_date, end\_date, status) **VALUES** (1,’computer’,20000,0,2025-05-27,2025-12-12);

根據使用者的ID跟輸入的資料去新增儲蓄目標，這樣輸入表示小宋新增一個儲蓄目標是computer，必須存到20000元，現在儲存的金額為0，時間從2025-05-27開始到2025-12-12結束，系統就會根據這些輸入資料去建立新的儲蓄目標，可以供使用者查看、完成目標。

1. VIEW表建立

我們會建立四種VIEW表分別是saving\_goal\_status、user\_transaction\_history、user\_financial\_summary、user\_category\_summary。以下會逐一講解。

* 1. saving\_goal\_status

可以查看資料庫內所有人的名字、開始結束儲蓄時間、目標金額、目前已存的金額以及狀態，之後可以再利用這個表來找出特定人有關於儲蓄目標的資料且能算出離目標還差多少金額、多少時間...等。

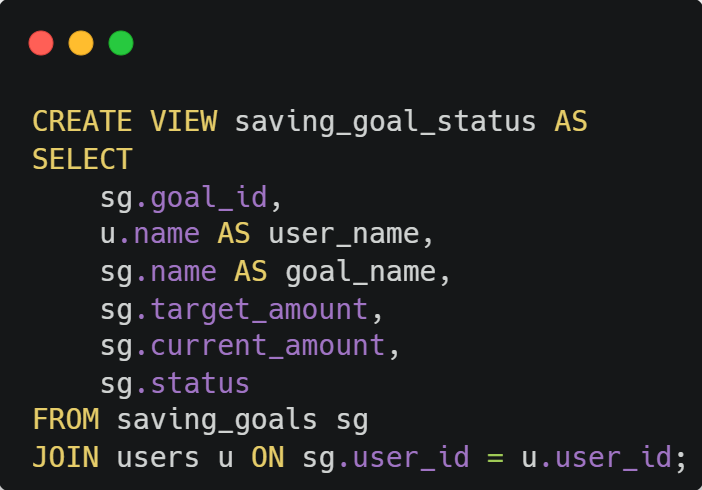


圖 12：saving\_goal\_status SQL

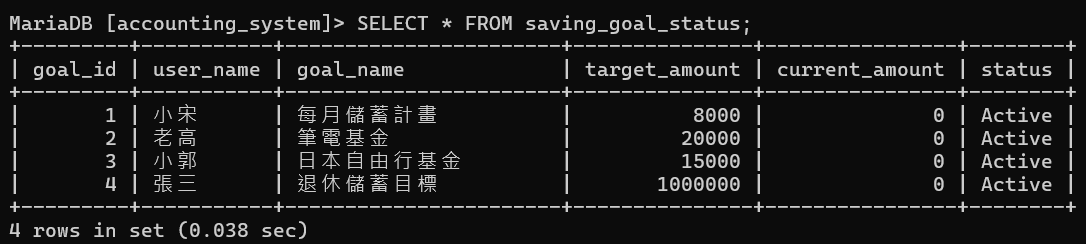


圖 13：saving\_goal\_status實際建置

* 1. user\_transaction\_history

可以查看資料庫內所有人的名字、每筆交易紀錄（含交易日期、金額、類別），之後也能透過此表查詢出想要的人的所有交易紀錄。

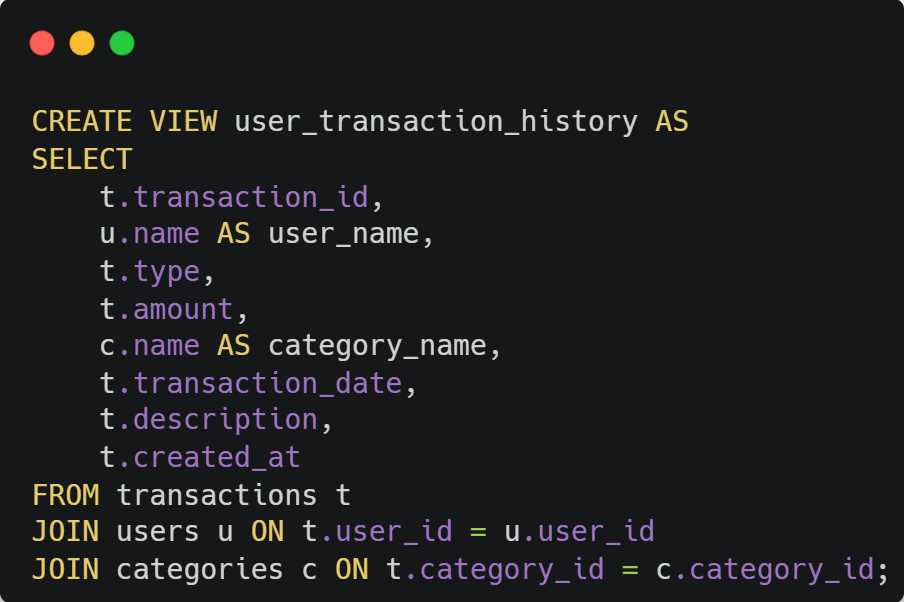


圖 14：user\_transaction\_history SQL

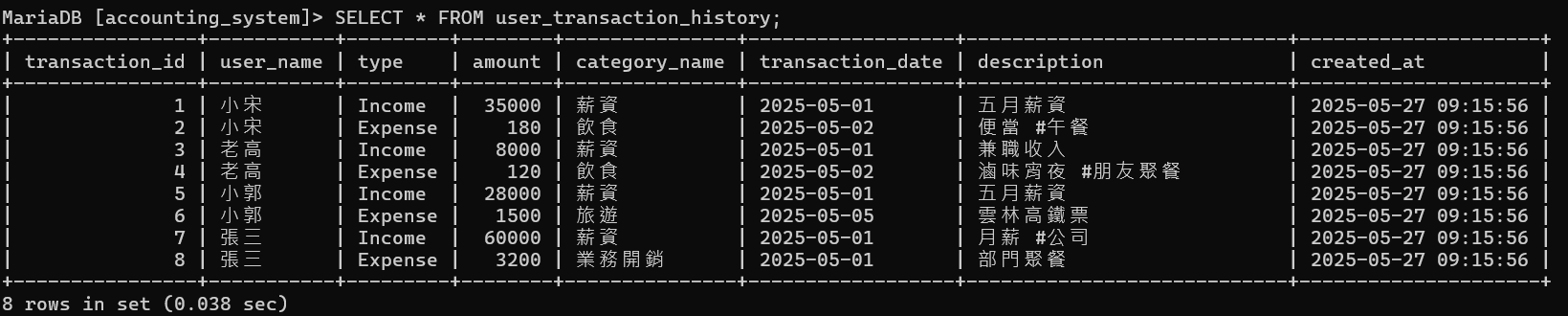


圖 15：user\_transaction\_history實際建置

* 1. user\_financial\_summary

可以查看資料庫內所有人的個人資料（含名字、總資產、創建時間、帳號），以及他交易的總支出含總花費，之後也能透過此表查詢出想要的人的個人資料。

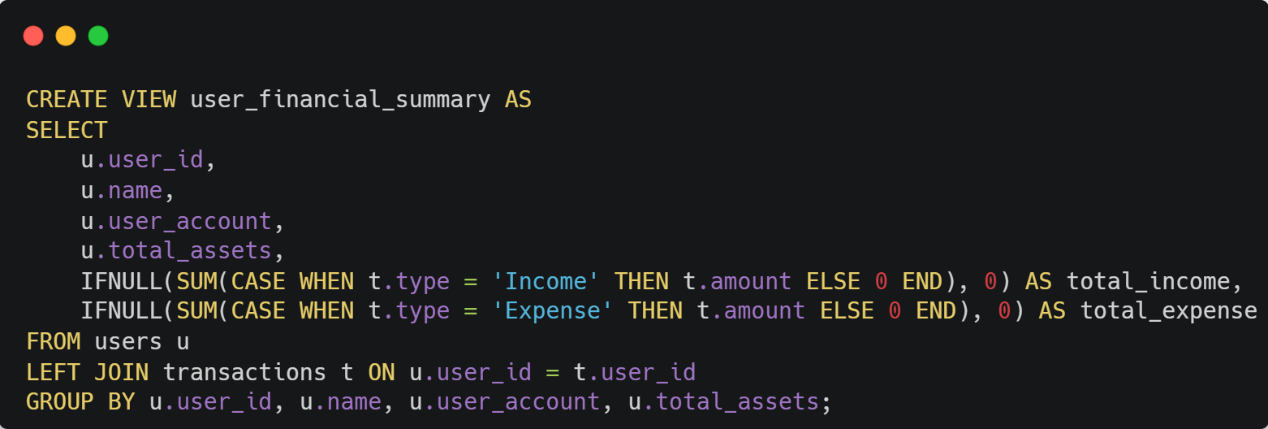


圖 16：user\_financial\_sunnary SQL

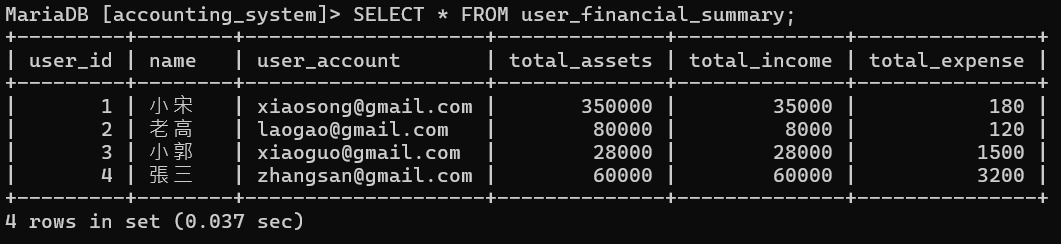


圖 17：user\_financial\_sunnary 實際建置

* 1. user\_category\_summary

可以查看資料庫內所有人每種分類的總收入以及總支出，所以會從資料庫拿每種分類下的收入以及支出做運算，以便之後實現能讓使用者查看個人帳戶下的所有分類的總收支。

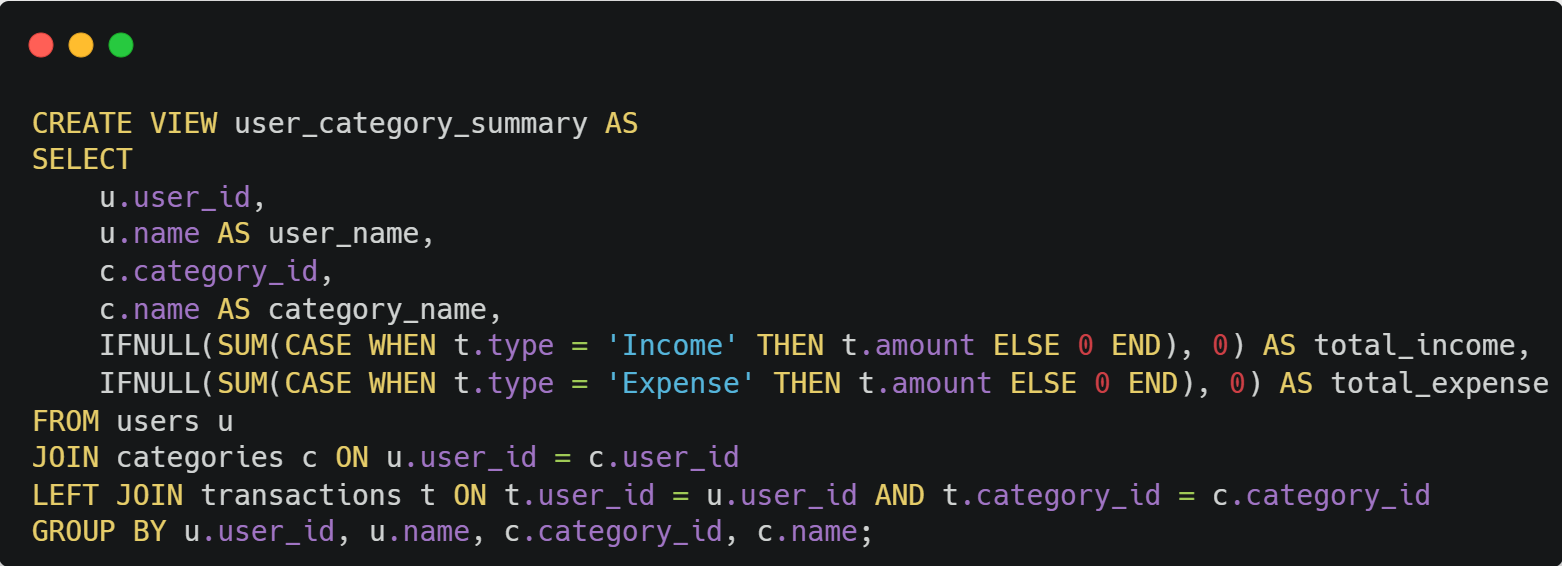


圖 18：user\_category\_summary SQL

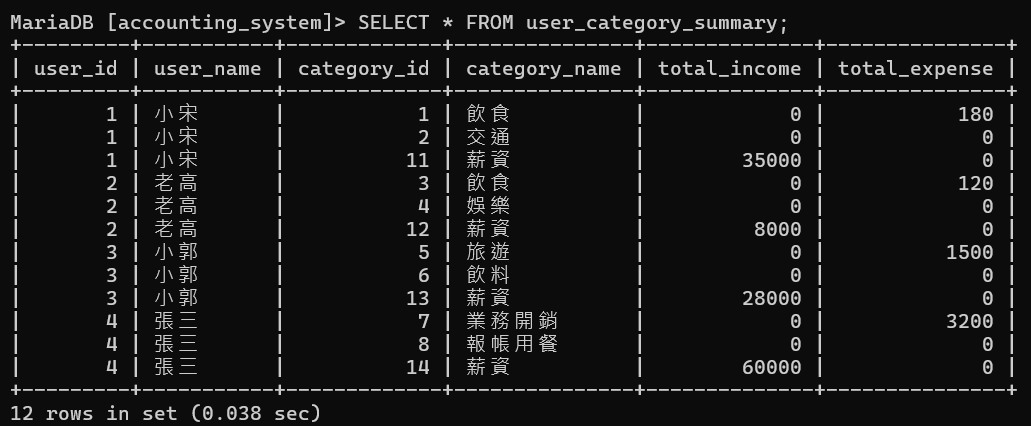


圖 19：user\_category\_summary實際建置

* **財務顯示方法**



圖 20：財務顯示畫面

登入後畫面上方會顯示該帳戶的總資產、總收入、總支出，方便使用者查看自己的所有紀錄，利用已經創建好的view來進行查詢與操作，使用user\_category\_summary的view，來抓取指定使用者的總資產、總收入、總支出。

* **財務查詢：**

**SELECT** total\_assets, total\_income, total\_expense **FROM** user\_category\_summary **WHERE** user\_id = 1;

因為是透過通用view進行查詢，所以需要加上限制條件當前是哪個使用者的user\_id，避免其他使用者的資料外洩，目前舉例的是user\_id為1的使用者他的總資產、總收入、總支出為多少，回傳之後可以讓系統顯示在GUI上，方便使用者去查看。

* **財務顯示方法**

1. 分工