Schema

```
CREATE TABLE "dept" (
    "dept_ID"     varchar(10),
    "dept_name"   varchar(30),
    "degree"      varchar(10),
    CONSTRAINT PK_dept PRIMARY KEY (dept_ID), CONSTRAINT CK_dept_dept_name_degree UNIQUE(dept_name,degree),
    CONSTRAINT VS_degree CHECK (degree IN ('學士','碩士','博士'))
);

CREATE TABLE "dept_class" (
    "dept_class_ID"     varchar(10),
    "dept_ID"           varchar(10),
    "admission_year"    INTEGER,
    "class"             varchar(1),
    CONSTRAINT PK_dept_class PRIMARY KEY (dept_class_ID), CONSTRAINT CK_dept_class_dept_ID_admission_year_class UNIQUE(dept_ID,admission_year,class),
    CONSTRAINT FK_dept_class_REF_dept_ID FOREIGN KEY (dept_ID) REFERENCES dept(dept_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT VS_admission_year CHECK (admission_year >= 4),
    CONSTRAINT VS_class CHECK (class IN ('A','B','C','D'))
);
CREATE TABLE "student" (
    "student_ID"        varchar(10),
    "student_name"      varchar(20),
    "cur_dept_class_ID" varchar(10),
    "cur_status"        varchar(10),
    CONSTRAINT PK_student PRIMARY KEY (student_ID),
    CONSTRAINT FK_student_REF_cur_dept_class_ID FOREIGN KEY (cur_dept_class_ID) REFERENCES dept_class(dept_class_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT VS_cur_status CHECK (cur_status IN ('在學','休學','退學','畢業'))
);
CREATE TABLE "teacher" (
    "teacher_ID"        varchar(10),
    "teacher_name"      varchar(20),
    CONSTRAINT PK_teacher PRIMARY KEY (teacher_ID)
);


CREATE TABLE "subject" (
    "subject_ID"        varchar(10),
    "subject_name"      varchar(255),
    "subject_credit"    INTEGER,
    "teacher_ID"   varchar(10),
    CONSTRAINT PK_subject PRIMARY KEY (subject_ID), CONSTRAINT CK_subject_subjecct_name_teacher_ID UNIQUE(subject_name,teacher_ID),
    CONSTRAINT FK_subject_REF_teacher_ID FOREIGN KEY (teacher_ID) REFERENCES teacher(teacher_ID) ON DELETE CASCADE ON UPDATE CASCADE,
);
CREATE TABLE "building" (
    "building_ID"       varchar(10),
    "building_name"     varchar(30),
    CONSTRAINT PK_building PRIMARY KEY (building_ID)
);


CREATE TABLE "room" (
    "room_ID"     varchar(10),
    "room_name"   varchar(30),
    "building_ID" varchar(10),
    "room_limit"  INT,
    CONSTRAINT PK_room PRIMARY KEY (room_ID), CONSTRAINT CK_room_room_name_building_ID UNIQUE(room_name,building_ID),
    CONSTRAINT FK_room_REF_building_ID FOREIGN KEY (building_ID) REFERENCES building(building_ID) ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE "course" (
    "course_ID"         varchar(10),
    "subject_ID"        varchar(10),
    "semester"          varchar(10),
    "room_ID"           varchar(10),
    "course_time"       varchar(200),
    "course_limit"      INTEGER,
    "course_status"     varchar(10),
    "course_is_online"  varchar(10),
    CONSTRAINT PK_course PRIMARY KEY (course_ID),CONSTRAINT CK_course_subject_ID_semester UNIQUE(subject_ID,semester),
    CONSTRAINT FK_course_REF_subject_ID FOREIGN KEY (subject_ID) REFERENCES subject(subject_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_course_REF_room_ID FOREIGN KEY (room_ID) REFERENCES room(room_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT VS_semester CHECK (semester LIKE '\[[1-9][0-9][0-9],[12]\]' {escape '\'}),
    CONSTRAINT VS_course_time CHECK (dbo.valid_time(course_time) > 0),
    CONSTRAINT VS_course_status CHECK (course_status IN ('授課','計算成績','歷史資料')),
    CONSTRAINT VS_course_is_online CHECK (((room_ID IS NOT NULL) AND course_is_online IN ('實體','並行')) OR course_is_online = '線上')
);
CREATE OR ALTER TRIGGER VS_course_limit ON course FOR INSERT, UPDATE AS BEGIN IF (ROWCOUNT_BIG() = 0) RETURN ELSE
    IF EXISTS (
        SELECT course_limit FROM course C INNER JOIN room R on (C.room_ID = R.room_ID) WHERE C.course_limit > R.room_limit
    ) ROLLBACK TRANSACTION
    ELSE RETURN;
END
CREATE OR ALTER TRIGGER room_use ON course FOR INSERT, UPDATE AS BEGIN IF (ROWCOUNT_BIG() = 0) RETURN ELSE
    IF EXISTS (
        SELECT C1.room_ID
        FROM course C1 JOIN course C2 ON( C1.course_ID != C2.course_ID AND C1.semester = C2.semester AND C1.room_ID = C2.room_ID )
        WHERE dbo.conflict_period(C1.course_time,C2.course_time) = 1
    ) ROLLBACK TRANSACTION
    ELSE RETURN;
END

CREATE TABLE "selection" (
    "course_ID"             varchar(10),
    "student_ID"            varchar(10),
    "select_dept_class_ID"  varchar(10),
    "select_type"           varchar(10),
    "select_result"         varchar(10),
    CONSTRAINT PK_select PRIMARY KEY (course_ID,student_ID),
    CONSTRAINT FK_select_REF_course_ID FOREIGN KEY (course_ID) REFERENCES course(course_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_select_REF_student_ID FOREIGN KEY (student_ID) REFERENCES student(student_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_select_REF_select_dept_class_ID FOREIGN KEY (select_dept_class_ID) REFERENCES dept_class(dept_class_ID),
    CONSTRAINT VS_select_type CHECK (select_type IN ('必修','選修','必選')),
    CONSTRAINT VS_select_result CHECK (select_result IN ('中選','落選'))
);
CREATE OR ALTER TRIGGER course_attend ON selection FOR INSERT, UPDATE AS BEGIN IF (ROWCOUNT_BIG() = 0) RETURN ELSE
    IF EXISTS (
        SELECT S1.student_ID
        FROM selection S1 JOIN selection S2 ON S1.student_ID = S2.student_ID AND S1.select_result = '中選' AND S2.select_result = '中選' AND S1.course_ID != S2.course_ID
            JOIN course C1 ON S1.course_ID = C1.course_ID JOIN course C2 ON S2.course_ID = C2.course_ID AND C1.semester = C2.semester
        WHERE dbo.conflict_period(C1.course_time,C2.course_time) = 1
    ) ROLLBACK TRANSACTION
    ELSE RETURN;
END

CREATE TABLE "score" (
    "course_ID"     varchar(10),
    "student_ID"    varchar(10),
    "student_score" NUMERIC(5,2),
    CONSTRAINT PK_score PRIMARY KEY (course_ID,student_ID),
    CONSTRAINT FK_score_REF_course_ID FOREIGN KEY (course_ID) REFERENCES course(course_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_score_REF_student_ID FOREIGN KEY (student_ID) REFERENCES student(student_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT VS_student_score CHECK (student_score >= 0 AND student_score <= 100)
);
CREATE OR ALTER TRIGGER have_score ON score FOR INSERT, UPDATE AS BEGIN IF (ROWCOUNT_BIG() = 0) RETURN ELSE
    IF EXISTS (
        SELECT * FROM inserted WHERE NOT EXISTS(
            SELECT * FROM selection S WHERE S.course_ID = inserted.course_ID AND S.student_ID = inserted.student_ID AND S.select_result = '中選'
        )
    ) ROLLBACK TRANSACTION
    ELSE RETURN;
END
CREATE TABLE "feedback" (
    "course_ID"     varchar(10),
    "student_ID"    varchar(10),
    "feedback_rank" INT,
    CONSTRAINT PK_feedback PRIMARY KEY (course_ID,student_ID),
    CONSTRAINT FK_feedback_REF_course_ID FOREIGN KEY (course_ID) REFERENCES course(course_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_feedback_REF_student_ID FOREIGN KEY (student_ID) REFERENCES student(student_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT VS_feedback_rank CHECK (feedback_rank >= 1 AND feedback_rank <= 5)
);
CREATE OR ALTER TRIGGER can_feedback ON feedback FOR INSERT, UPDATE AS BEGIN IF (ROWCOUNT_BIG() = 0) RETURN ELSE
    IF EXISTS (
        SELECT * FROM inserted WHERE NOT EXISTS(
            SELECT * FROM selection S WHERE S.course_ID = inserted.course_ID AND S.student_ID = inserted.student_ID AND S.select_result = '中選'
        )
    ) ROLLBACK TRANSACTION
    ELSE RETURN;
END
```

dept

dept_ID	dept_name	degree
系所 ID	系所名稱	系所學位
varchar(10)	varchar(30)	varchar(10) ∈ {學士,碩士,博士}

UNIQUE(dept_name,degree)

dept_class

dept_class_ID	dept_ID	admission_year	class
系級班別 ID	所屬系所 ID	班級成立年度	班別
varchar(10)	varchar(10)	int ≥ 4 (1915)	varchar(1) ∈ {A,B,C,D}

UNIQUE(dept_ID,admission_year,class)

註：這邊還有一點報告時忘了跟教授討論，同樣是數學系 A 班，可能每一年都有一班，如果把 admission_year 移走，就只剩下 dept_class_ID 可以區別，但實際上從資料庫的角度 ID 並沒有這層意義，甚至也沒有任何實際意義。(這應該也是讓教授覺得奇怪，其他組如第七組設計了有意義的 ID，query 時卻不用那些意義來當條件的原因。因為那只是你 assign ID 一個大致，或說模糊的規律，並不是資料庫的 constraint，除非你特別也把它寫進去。而 assign 單欄 PK 是方便其他表 reference 過來，這點報告時有討論到。)

student

student_ID	student_name	cur_dept_class_ID	cur_status
學生 ID	學生姓名	學生目前系級班別 ID	學生目前狀態
varchar(10)	varchar(20)	varchar(10)	varchar(10) ∈ {在學,休學,退學,畢業}

teacher

teacher_ID	teacher_name
教師 ID	教師姓名
varchar(10)	varchar(20)

subject

subject_ID	subject_name	subject_credit	teacher_ID
課程 ID	課程名稱	課程學分數	授課教師 ID
varchar(10)	varchar(255)	int	varchar(10)

UNIQUE(subject_name,teacher_ID)

building

building_ID	building_name
館舍 ID	館舍正式名稱
varchar(10)	varchar(30)

room

room_ID	room_name	building_ID	room_limit
教室 ID	教室編號	所在館舍 ID	教室學生容量
varchar(10)	varchar(30)	varchar(10)	int

UNIQUE(room_name,building_ID)

course

course_ID	subject_ID	semester	room_ID	course_time	course_limit	course_status	course_is_online
授課紀錄 ID	課程 ID	授課學期	使用教室 ID	上課時間	修課人數限制	目前授課狀態	是否線上教學
varchar(10)	varchar(10)	json	varchar(10)	json	int	varchar(10) ∈ {授課,計算成績,歷史資料}	varchar(10) ∈ {實體,線上,並行}

UNIQUE(subject_ID,semester)

1. 會檢查兩個 json attributes 的格式(詳情後面解釋)
2. 若為全面線上教學課程允許不使用教室
3. 修課人數限制不能超出教室人數限制(若有使用)
4. 同一學期使用同一教室的課程不能衝堂(詳情後面解釋)
5. 教師授課不能衝堂(詳情後面解釋)

```
CREATE OR ALTER TRIGGER course_teach ON course FOR INSERT, UPDATE AS BEGIN
    IF (ROWCOUNT_BIG() = 0)
        RETURN;
    ELSE IF EXISTS (
        SELECT S1.teacher_ID
        FROM course C1 JOIN course C2 ON C1.course_ID != C2.course_ID AND C1.semester = C2.semester
        JOIN subject S1 ON C1.subject_ID = S1.subject_ID JOIN subject S2 ON C2.subject_ID = S2.subject_ID AND S1.teacher_ID = S2.teacher_ID
        WHERE dbo.conflict_period(C1.course_time,C2.course_time) = 1
    ) ROLLBACK TRANSACTION
    ELSE
        RETURN;
END
```

selection

course_ID	student_ID	select_dept_class_ID	select_type	select_result
授課紀錄 ID	學生 ID	學生選課時系級班別 ID	選課類別	選課最終結果
varchar(10)	varchar(10)	varchar(10)	varchar(10) ∈ {必修,選修,必選}	varchar(10) ∈ {中選,落選}

1. 學生選課不能衝堂(詳情後面解釋)
2. 學生不能未入學就選課

```
CREATE OR ALTER TRIGGER admission_before ON selection FOR INSERT, UPDATE AS BEGIN
    IF (ROWCOUNT_BIG() = 0)
        RETURN;
    ELSE IF EXISTS (
        SELECT S.student_ID
        FROM selection S JOIN dept_class D ON S.select_dept_class_ID = D.dept_class_ID JOIN course C ON S.course_ID = C.course_ID
        WHERE D.admission_year > dbo.get_year(C.semester)
    ) ROLLBACK TRANSACTION
    ELSE
        RETURN;
END
```

score

course_ID	student_ID	student_score
授課紀錄 ID	被評量學生 ID	分數
varchar(10)	varchar(10)	numeric(0.00 ~ 100.00)

1. 有相關的選課中選紀錄才能登記成績紀錄

feedback

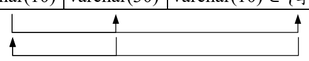
course_ID	student_ID	feedback_rank
授課紀錄 ID	回饋學生 ID	教學評量回饋分數
varchar(10)	varchar(10)	int ∈ {1,2,3,4,5}

1. 有相關的選課中選紀錄才能登記教學評量回饋分數紀錄

Functional Dependency

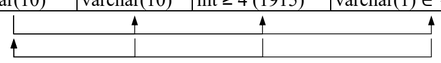
dept

dept_ID	dept_name	degree
系所 ID	系所名稱	系所學位
varchar(10)	varchar(30)	varchar(10) ∈ {學士,碩士,博士}



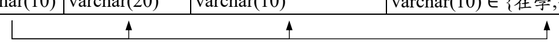
dept_class

dept_class_ID	dept_ID	admission_year	class
系級班別 ID	所屬系所 ID	入學年度	班別
varchar(10)	varchar(10)	int ≥ 4 (1915)	varchar(1) ∈ {A,B,C,D}



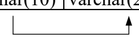
student

student_ID	student_name	cur_dept_class_ID	cur_status
學生 ID	學生姓名	學生目前系級班別 ID	學生目前狀態
varchar(10)	varchar(20)	varchar(10)	varchar(10) ∈ {在學,休學,退學,畢業}



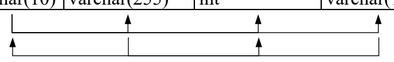
teacher

teacher_ID	teacher_name
教師 ID	教師姓名
varchar(10)	varchar(20)



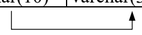
subject

subject_ID	subject_name	subject_credit	teacher_ID
課程 ID	課程名稱	課程學分數	授課教師 ID
varchar(10)	varchar(255)	int	varchar(10)



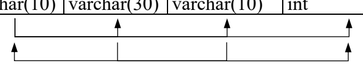
building

building_ID	building_name
館舍 ID	館舍正式名稱
varchar(10)	varchar(30)



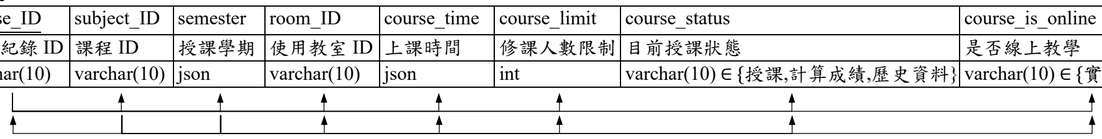
room

room_ID	room_name	building_ID	room_limit
教室 ID	教室編號	所在館舍 ID	教室學生容量
varchar(10)	varchar(30)	varchar(10)	int



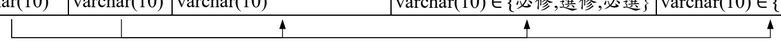
course

course_ID	subject_ID	semester	room_ID	course_time	course_limit	course_status	course_is_online
授課紀錄 ID	課程 ID	授課學期	使用教室 ID	上課時間	修課人數限制	目前授課狀態	是否線上教學
varchar(10)	varchar(10)	json	varchar(10)	json	int	varchar(10) ∈ {授課,計算成績,歷史資料}	varchar(10) ∈ {實體,線上,並行}



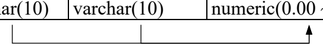
selection

course_ID	student_ID	select_dept_class_ID	select_type	select_result
授課紀錄 ID	學生 ID	學生選課時系級班別 ID	選課類別	選課最終結果
varchar(10)	varchar(10)	varchar(10)	varchar(10) ∈ {必修,選修,必選}	varchar(10) ∈ {中選,落選}



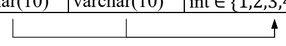
score

course_ID	student_ID	student_score
授課紀錄 ID	被評量學生 ID	分數
varchar(10)	varchar(10)	numeric(0.00 ~ 100.00)



feedback

course_ID	student_ID	feedback_rank
授課紀錄 ID	回饋學生 ID	教學評量回饋分數
varchar(10)	varchar(10)	int ∈ {1,2,3,4,5}



額外實務：json

在 json 的應用上，我們以 User Defined Function 包裝，透過 UDF 建構 interface 將 json 字串的格式及結構全部封鎖到資料庫端，隔離於 query 之外。

例如較簡單的授課學期，110 下學期格式為'[110,2]'，雖然「陣列[0]是學年度；陣列[1]是上下學期」這樣的概念對資料庫使用者很不直觀，而且也完全沒有體現出資料本身的意義，但我們可以運用 UDF 將這些莫名其妙的東西隔離出去：

```
ALTER FUNCTION [dbo].[semester_to_json] (@semester VARCHAR(10)) RETURNS VARCHAR(10) AS BEGIN
	RETURN ('['+SUBSTRING(@semester,1,LEN(@semester)-1)+','+SUBSTRING(@semester,LEN(@semester),1)+']')
END
ALTER FUNCTION [dbo].[get_year] (@semester VARCHAR(10)) RETURNS VARCHAR(10) AS BEGIN
	RETURN JSON_VALUE(@semester,'$[0]')
END
ALTER FUNCTION [dbo].[get_half] (@semester VARCHAR(10)) RETURNS VARCHAR(10) AS BEGIN
	RETURN JSON_VALUE(@semester,'$[1]')
END
ALTER FUNCTION [dbo].[get_semester] (@semester VARCHAR(10)) RETURNS VARCHAR(10) AS BEGIN
	RETURN dbo.get_year(@semester)+dbo.get_half(@semester)
END
```

光看這樣可能還覺得沒什麼，但那是因為學期 json 結構設計得很簡單，對於上課時間就沒有那麼輕鬆了。我們設計的上課時間 json 格式以「一 34,五 4」為例是'{"一":"34","五":"4"}'，因為只有運用 key-value 結構才能較好把上課時間的結構性表現出來。

雖然 sql 有提供 parse json 的 function，但要運用這些基本的 function 達到我們對於「上課時間」這個 attribute 的期待跟需求，需要進行很複雜的操作，這些操作如果全部塞進 query 讓人無法想像。這時就體現出用 UDF 建構 interface 的重要性了：

將前端格式轉換成 json 字串(過程中「星期七」正規化為「星期日」)

```
ALTER FUNCTION [dbo].[course_time_to_json] (@course_time VARCHAR(20))
RETURNS VARCHAR(200) AS BEGIN
	DECLARE @ret VARCHAR(200) = '{'
	DECLARE @tmp VARCHAR(20)
	DECLARE cur CURSOR LOCAL FOR
		SELECT value FROM STRING_SPLIT(@course_time, ',')
	OPEN cur
	FETCH NEXT FROM cur INTO @tmp
	WHILE @@FETCH_STATUS = 0 BEGIN
		IF(SUBSTRING(@tmp,1,1) = '七')
			SET @ret += ""+'日'+"":"+SUBSTRING(@tmp,2,LEN(@tmp)-1)+""
		ELSE
			SET @ret += ""+SUBSTRING(@tmp,1,1)+"":"+SUBSTRING(@tmp,2,LEN(@tmp)-1)+""
		FETCH NEXT FROM cur INTO @tmp
		IF(@@FETCH_STATUS = 0)
			SET @ret += ','
		ELSE
			SET @ret += '}'
	END
	CLOSE cur
	DEALLOCATE cur
	RETURN @ret
END
```

取得前端格式的上課時間字串(過程中「星期七」正規化為「星期日」)

```
ALTER FUNCTION [dbo].[get_time] (@time VARCHAR(200))
RETURNS VARCHAR(135) AS BEGIN
	DECLARE @ret VARCHAR(135) = ''
	DECLARE @tmp VARCHAR(20)
	SET @tmp = JSON_VALUE(@time,'$."一"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',一'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."二"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',二'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."三"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',三'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."四"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',四'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."五"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',五'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."六"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',六'+@tmp
	SET @tmp = JSON_VALUE(@time,'$."七"')
	IF(@tmp IS NOT NULL)
		SET @ret += ',日'+@tmp
	ELSE BEGIN
		SET @tmp = JSON_VALUE(@time,'$."日"')
		IF(@tmp IS NOT NULL)
			SET @ret += ',日'+@tmp
	END
	RETURN SUBSTRING(@ret,2,LEN(@ret))
END
```

取得由上課星期構成的 table

```
ALTER FUNCTION [dbo].[get_days] (@time VARCHAR(200))
RETURNS @ret TABLE(days VARCHAR(2)) AS BEGIN
	DECLARE @all_days TABLE (days VARCHAR(2))
	INSERT INTO @all_days VALUES
	('一'),('二'),('三'),('四'),('五'),('六'),('七'),('日')
	INSERT INTO @ret
	SELECT days
	FROM @all_days
	WHERE JSON_VALUE(@time,'$."'+days+'"') IS NOT NULL
	RETURN
END
```

取得由上課星期跟時段兩欄構成的 table

```
ALTER	FUNCTION [dbo].[get_period] (@time VARCHAR(200))
RETURNS @ret TABLE(days VARCHAR(2),period VARCHAR(1)) AS BEGIN
	DECLARE @all_days TABLE (days VARCHAR(2))
	INSERT INTO @all_days
	SELECT days FROM get_days(@time)
	DECLARE @all_period TABLE (period VARCHAR(1))
	INSERT INTO @all_period VALUES
	('1'),('2'),('3'),('4'),('Z'),('5'),('6'),('7'),('8'),('9'),('A'),('B'),('C'),('D'),('E'),('F')
	INSERT INTO @ret
	SELECT TOP(200) days,period
	FROM @all_days,@all_period
	WHERE CHARINDEX(period,JSON_VALUE(@time,'$."'+days+'"')) > 0
	ORDER BY days
	RETURN
END
```

測試是否在星期幾有課(搜尋目標也是 json 的版本)

```
ALTER FUNCTION [dbo].[on_days_json] (@time VARCHAR(200),@target VARCHAR(200))
RETURNS BIT AS BEGIN
	IF(EXISTS(SELECT days FROM get_days(@target) EXCEPT SELECT days FROM get_days(@time)))
		RETURN 0
	RETURN 1
END
```

測試是否在星期幾有課(搜尋目標字串隨便你打的版本)

```
ALTER FUNCTION [dbo].[on_days] (@time VARCHAR(200),@target VARCHAR(200))
RETURNS BIT AS BEGIN
	IF(CHARINDEX('一',@target) > 0 AND CHARINDEX('一',@time) <= 0 )
		RETURN 0
	IF(CHARINDEX('二',@target) > 0 AND CHARINDEX('二',@time) <= 0 )
		RETURN 0
	IF(CHARINDEX('三',@target) > 0 AND CHARINDEX('三',@time) <= 0 )
		RETURN 0
	IF(CHARINDEX('四',@target) > 0 AND CHARINDEX('四',@time) <= 0 )
		RETURN 0
	IF(CHARINDEX('五',@target) > 0 AND CHARINDEX('五',@time) <= 0 )
		RETURN 0
	IF(CHARINDEX('六',@target) > 0 AND CHARINDEX('六',@time) <= 0 )
		RETURN 0
	IF((CHARINDEX('七',@target) > 0 OR CHARINDEX('日',@target) > 0) AND CHARINDEX('日',@time) <= 0)
		RETURN 0
	RETURN 1
END
```

測試是否在某些時段有上課(學生系統上篩選搜尋常需要)

```
ALTER	FUNCTION [dbo].[on_period] (@time VARCHAR(200),@target VARCHAR(200))
RETURNS BIT AS BEGIN
	IF(dbo.on_days_json(@time,@target) <= 0)
		RETURN 0
	ELSE IF(EXISTS(SELECT days,period FROM get_period(@target) EXCEPT SELECT days,period FROM get_period(@time)))
		RETURN 0
	RETURN 1
END
```

最後最重要的測試是否衝堂

```
ALTER FUNCTION [dbo].[conflict_period] (@time1 VARCHAR(200),@time2 VARCHAR(200))
RETURNS BIT AS BEGIN
	IF(EXISTS(SELECT days,period FROM get_period(@time1) INTERSECT SELECT days,period FROM get_period(@time2)))
		RETURN 1
	RETURN 0
END
```

Schema 那邊動用了很多 trigger 來對資料庫的資料作限制，其中就包含各種情況的課程衝堂，用的就是這邊的 UDF。另外，因為資料庫支援 json 的方式，是用 parse function 把單純的字串資料當成 json 看，換句話說只有你 parse 的時候它是 json，其他時候它都是單純的字串，這就意味著你也不能保證使用者放進去的資料真的是 json。因此我們也希望確保這部分存入資料庫的資料「在某種程度上」符合我們制訂的格式，對於簡單的「學期」可以用 LIKE，但因為 LIKE 的表達力差 Regular Exp 一大截，到了「上課時間」已經不管用，此時這些 UDF 也派上用場：

```
ALTER FUNCTION [dbo].[valid_time] (@time VARCHAR(200))
RETURNS BIT AS BEGIN
	IF(ISJSON(@time) <= 0)
		RETURN 0
	ELSE IF(NOT EXISTS (SELECT * FROM get_period(@time)))
		RETURN 0
	RETURN 1
END
```

只要它至少是個 json，然後我們能 get_period，就視為勉強堪用。

雖然這次出的題目完全沒有要動到這部分，講師也不諱言因為這樣結構性的資料基本上很難用正規化處理，也不確定大家能不能使用 json 的方式，所以出題時他刻意迴避了，但因為 mssql 剛好有支援，我們就寫了。

這些 UDF 徹底將前後端分離，理論上使用者可以在完全不知道 json 架構的情況下，用這些 UDF 插入、刪除、更新資料，並直觀簡明地在 query 中對這些 json 格式的欄位做出有符合欄位資料意義的分析跟條件判斷，對使用者而言學期資料還是'1102'；上課時間還是'一 34,五 4'，而且本來資料庫無法支援的功能，現在不僅能直接放在 query 當中，還都只要放一小段 function call。

本來希望 DBMS 自動幫你限制課程不能衝堂，在時間都是字串資料的情況下應該是不可能的，現在弄成 json 容納了資料本身的結構性，又搭配 UDF 模組化對資料的結構式分析，就做得到了。

額外實務：別名問題

我們知道很多具體的 entity 都會有別名的問題，例如講師就說了「曹操跟曹孟德到底是不是同一個人」的問題，這部分在校務行政系統這樣比較嚴肅正式的應用上，因為通常都只看本名，比較不會有，但是像館舍或是課名，別名問題就很嚴重了。

例如「工五」、「工程五館」、「資工系館」指的都是同一棟館舍，而資料庫正規化其實有一部分也是為了解決別名問題。但這種方法是「禁止使用別名」，而我們發現了一個很有趣的方法，可以把別名也整合進資料庫，讓使用者「合法」使用別名。

當然不是真的在資料庫裡合法，看一下這張 table：

```
CREATE TABLE "building_ID_assign" (
        "name"        varchar(30),
        "ID" varchar(10),
        CONSTRAINT PK_building_ID_assign PRIMARY KEY (name)
);
```

name	ID
文學一館	A
文學二館	C2
工程一館	E
工程二館	E1
電機館	E1
工程三館	E2
機械館	E2
環工化工館	E3
環工館	E3
工程四館	E3
機電實驗室	E4
大型力學實驗室	E5
工五	E6
工程五館	E6
工程五館大樓	E6
資工系館	E6
理學院教學館	H2
普化實驗大樓	H2
客家學院大樓	HK
管理一管	I
志希館	I
管理二館	I1
國鼎光電大樓	IL
國鼎圖書資料館	L3
黑盒子	LS
黑盒子劇場	LS
文學三館	LS
台灣電影研究中心	LS
人文社會科學大樓	LS
鴻經館	M
綜教館	O
太空及遙測研究中心	R2
研究中心大樓二期	R3
科學一館	S
科學二館	S1
科學三館	S2
科學四館	S4
健雄館	S4
科學五館	S5
教研大樓	TR
教學研究綜合大樓暨大禮堂	TR
依仁堂	YH
體育館	YH

注意我們把 PK 設在館舍的名稱而不是 ID。有了這張表，我們再搭配一個 User Defined Procedure 用來設定新別名：

```
ALTER PROCEDURE [dbo].[set_building_alias]
@name1 VARCHAR(30), @name2 VARCHAR(30), @result BIT output AS BEGIN
    DECLARE @ID VARCHAR(10)
        = (SELECT ID FROM building_ID_assign WHERE name = @name1)
    SET @result = 0
    IF(@ID IS NOT NULL) BEGIN
        IF(NOT EXISTS(
        SELECT ID FROM building_ID_assign WHERE name = @name2)) BEGIN
            INSERT INTO building_ID_assign VALUES (@name2,@ID)
            SET @result = 1
        END
    END ELSE BEGIN
        SET @ID = (SELECT ID FROM building_ID_assign WHERE name = @name2)
        IF(@ID IS NOT NULL) BEGIN
            INSERT INTO building_ID_assign VALUES (@name1,@ID)
            SET @result = 1
        END
    END
END
```

跟一個 UDF 當 interface：

```
ALTER FUNCTION [dbo].[get_building_ID] (@name VARCHAR(30))
RETURNS VARCHAR(10) AS BEGIN
    RETURN (SELECT ID FROM building_ID_assign WHERE name = @name)
END
```

這樣就容納了別名。舉例來說，「曾經在工五上過課的課程」：

```
SELECT DISTINCT S.subject_name
FROM course C
    JOIN subject S ON C.subject_ID = S.subject_ID
    JOIN room R ON C.room_ID = R.room_ID
WHERE R.building_ID = dbo.get_building_ID('工五')
```

跟「曾經在資工系館上過課的課程」：

```
SELECT DISTINCT S.subject_name
FROM course C
    JOIN subject S ON C.subject_ID = S.subject_ID
    JOIN room R ON C.room_ID = R.room_ID
WHERE R.building_ID = dbo.get_building_ID('資工系館')
```

看起來你是用不同的名字下 query，但過了 interface 之後，因為拿到了一樣的 ID，所以其實是一樣的 query，也符合正規化。

額外實務：歷史資料匯入

理想上我們應該在最一開始就把 schema 設計好，但不完備定理告訴我們，不論你想得再全面再周到，永遠存在你沒想到的東西，只是你還沒發現它。一個很好的例子就是近年因為疫情遠距教學這個新概念浮出檯面，在幾年前根本不可能有人會想到需要「course_is_online」這個 attribute。

那麼如果是已經辦學幾十年的學校，當他的校務系統資料庫 schema 終於完全不堪使用的那一天，難道為了用新的 schema 就要放棄數十載的歷史嗎？

在做專題的過程中我們發現，在對新舊 schema 同時都有深度理解的情況下，可以根據兩者的連結精細地打造 query，把舊 schema 下的資料轉移到新 schema。

例如要把舊資料學生的部分匯入到 student，我們是這樣做的：

```
INSERT INTO student (student_ID, student_name, cur_dept_class_ID, cur_status)
SELECT DISTINCT ID, student_name, dept_class_ID, student_status
FROM (origin LEFT OUTER JOIN student_ID_assign ON student_name = name)
    LEFT OUTER JOIN (dept_class JOIN dept ON dept_class.dept_ID = dept.dept_ID)
    ON (student_dept = dept_name AND student_class = class
        AND CAST(SUBSTRING(semester,1,LEN(semester)-1) AS INT)+1-student_grade = admission_year)
```

origin 就是舊資料表，然後 student_ID_assign 就是像上面別名問題那邊的表，給學生學號用的(對這就是我們會發現那種方法的原因)。

當然這也有一些問題，例如數學系根本不知道是大學部還是碩博班，在新 schema 當中系名不是唯一值，同樣數學系可以有學士班、碩士班、博士班，此時你只能先當成舊資料中數學系都是大學生，然後你先在 dept 放上大學部的數學系，還不能放碩博班因為匯入的時候 JOIN 會有問題，這邊匯入完歷史資料之後，再回去加上碩博班的數學系。像這樣的情況，在舊 schema 中你不知道數學系的學生哪些是大學部；哪些是碩博班，你只是「沒有資訊」，但到了新的 schema，你預設從舊資料來的數學系學生都是大學部，你是「拿了錯誤資訊」。哪個比較嚴重？相信教授一定會說是錯誤資訊，這就是因為新 schema 要記錄比舊 schema 更多的資訊，把舊資料整合過去時會面對的問題。

又例如匯入到 course，是這樣做的：

```
INSERT INTO course (course_ID,subject_ID,semester,room_ID,course_time,course_limit,course_status,course_is_online)
SELECT DISTINCT course_no+semester,course_no,dbo.semester_to_json(semester),
room_ID,dbo.course_time_to_json(course_time),course_limit,'歷史資料','實體'
FROM origin O LEFT OUTER JOIN teacher T ON O.teacher_name = T.teacher_name
    LEFT OUTER JOIN room ON course_room = room_name
WHERE course_is_online = '否'

INSERT INTO course (course_ID,subject_ID,semester,room_ID,course_time,course_limit,course_status,course_is_online)
SELECT DISTINCT course_no+semester,course_no,dbo.semester_to_json(semester),
    room_ID,dbo.course_time_to_json(course_time),course_limit,'歷史資料','線上'
FROM origin O LEFT OUTER JOIN teacher T ON O.teacher_name = T.teacher_name
    LEFT OUTER JOIN room ON course_room = room_name
WHERE course_is_online = '是'
```

至於 selection 就很頭痛了，新 schema 明確限制學生選課不能衝堂，但原始資料裡面一堆人中選都衝堂！這種新 schema 限制比舊 schema 嚴格的情況，就只能放棄那些違反新 schema constraint 的資料了，像這樣：

```
CREATE OR ALTER VIEW intended_selection AS
SELECT DISTINCT course_ID,student_ID,cur_dept_class_ID,course_type,select_result
FROM origin O LEFT OUTER JOIN student S ON O.student_name = S.student_name
    LEFT OUTER JOIN (course C JOIN subject Sub ON C.subject_ID = Sub.subject_ID)
    ON (O.course_name = Sub.subject_name AND C.semester = dbo.semester_to_json(O.semester))
GO
CREATE OR ALTER VIEW invalid_select_student AS
SELECT S1.student_ID
FROM intended_selection S1 JOIN intended_selection S2
    ON S1.student_ID = S2.student_ID AND S1.select_result = '中選' AND S2.select_result = '中選' AND S1.course_ID != S2.course_ID
    JOIN course C1 ON S1.course_ID = C1.course_ID JOIN course C2 ON S2.course_ID = C2.course_ID AND C1.semester = C2.semester
WHERE dbo.conflict_period(C1.course_time,C2.course_time) = 1
GO
INSERT INTO selection (course_ID,student_ID,select_dept_class_ID,select_type,select_result)
SELECT *
FROM intended_selection intended
WHERE intended.student_ID NOT IN (SELECT * FROM invalid_select_student)
```

解題

1. 1102 學期的「A0001 微積分」，因故上課地點要由 K205 修改到 K210 大教室，該怎麼做？

註：後來發現第三種答案是錯的。或者至少可以說是會把所有 1102 學期的微積分全部改到 K210，取決於題目的 A0001 作何解讀，也可以勉強算另一種答案啦……

1.sql - DESKTOP-6...ourse (mssql (54)) | 3.sql - DESKTOP-6...ourse (mssql (53))

```
--UPDATE course SET room_ID = 'K210'
--WHERE course_ID = 'A00011102'

UPDATE course SET room_ID = 'K210'
 WHERE subject_ID = 'A0001' AND dbo.get_semester(semester) = '1102'

--UPDATE course SET room_ID = 'K210'
--FROM course C JOIN subject S ON (C.subject_ID = S.subject_ID)
--WHERE subject_name = '微積分' AND dbo.get_semester(semester) = '1102'
```

118 %

訊息

```
(1 個資料列受到影響)

完成時間: 2022-05-27T18:56:43.6869804+08:00
```

2. 請列出 1102 學期的「A0002 計算機概」的修課名單（點名表）。

2.sql - DESKTOP-6...ourse (mssql (54)) | SQLQuery24.sql -...ourse (mssql (53))*

```
CREATE OR ALTER VIEW list AS
SELECT student_ID
FROM selection S JOIN course C ON S.course_ID = C.course_ID
WHERE subject_ID = 'A0002' AND dbo.get_semester(semester) = '1102'
    AND select_result = '中選'
GO
SELECT ST.student_ID ID,ST.student_name name,D.dept_name dept,
    (110-DC.admission_year+1) grade,DC.class,ST.cur_status
FROM student ST JOIN dept_class DC ON cur_dept_class_ID = dept_class_ID
    JOIN dept D ON DC.dept_ID = D.dept_ID
WHERE ST.student_ID IN (SELECT student_ID FROM list)
ORDER BY CASE
    WHEN ST.cur_status = '在學' THEN 1
    WHEN ST.cur_status = '休學' THEN 2
    WHEN ST.cur_status = '退學' THEN 3
END
```

118 %

結果 | 訊息

	ID	name	dept	grade	class	cur_status
1	110201505	趙雷	數學系	1	A	在學
2	110201507	夏侯惇	數學系	1	A	在學
3	110522002	華陀	資訊工程研究所	1	A	在學
4	110502502	龐統	資訊工程系	1	A	休學
5	110522001	華雄	資訊工程研究所	1	A	退學

3. 請列出 1102 學期，成績不及格的修課學生資料（大學部：低於 60 分、碩博：70 分）。

3.sql - DESKTOP-6...ourse (mssql (53))

```
SELECT ST.student_ID ID,student_name name,SUB.subject_name subject,SC.student_score score
FROM score SC JOIN course CO ON SC.course_ID = CO.course_ID
    JOIN subject SUB ON CO.subject_ID = SUB.subject_ID
    JOIN student ST ON SC.student_ID = ST.student_ID
    JOIN selection SE ON (SC.course_ID = SE.course_ID AND SC.student_ID = SE.student_ID)
    JOIN dept_class DC ON SE.select_dept_class_ID = dept_class_ID
    JOIN dept D ON DC.dept_ID = D.dept_ID
WHERE dbo.get_semester(CO.semester) = '1102'
    AND ((D.degree = '學士' AND SC.student_score < 60)
    OR (D.degree IN ('碩士','博士') AND SC.student_score < 70))
ORDER BY student_name,score
```

118 %

結果 | 訊息

	ID	name	subject	score
1	110502504	甘寧	統計學	46.00
2	110522006	呂布	虛擬實境	65.00
3	110522002	華陀	演算法	60.00
4	110201505	趙雷	計算機概論	49.00
5	110522003	劉備	音樂欣賞	56.00
6	110522005	諸葛亮	音樂欣賞	55.00
7	110201506	關興	微積分	55.00

4. 以中選比例（中選人次 / 加選人次 x 100）推測 1102 學期受學生歡迎的熱門加選課程？

4.sql - DESKTOP-6...ourse (mssql (54)) | SQLQuery24.sql -...ourse (mssql (53))*

```
CREATE OR ALTER VIEW select_AGG AS
SELECT P.ID,enroll_count,select_count
FROM(
SELECT S.course_ID ID,COUNT(*) enroll_count
FROM selection S JOIN course C ON S.course_ID = C.course_ID
WHERE dbo.get_semester(C.semester) = '1102' AND select_result = '中選'
GROUP BY S.course_ID
) G JOIN (
SELECT S.course_ID ID,COUNT(*) select_count
FROM selection S JOIN course C ON S.course_ID = C.course_ID
WHERE dbo.get_semester(C.semester) = '1102'
GROUP BY S.course_ID
) P ON G.ID = P.ID
GO
SELECT subject_name,T.teacher_name,enroll_count,select_count,
    CAST(CAST(enroll_count AS real)/select_count AS NUMERIC(3,2)) AC_rate
FROM select_AGG AGG JOIN course C on AGG.ID = C.course_ID
    JOIN subject SUB ON C.subject_ID = SUB.subject_ID
    JOIN teacher T ON SUB.teacher_ID = T.teacher_ID
ORDER BY AC_rate
```

118 %

結果 | 訊息

	subject_name	teacher_name	enroll_count	select_count	AC_rate
1	經濟學	孔丘	3	4	0.75
2	音樂欣賞	巴哈	10	13	0.77
3	虛擬實境	劉邦	6	7	0.86
4	演算法	達文西	3	3	1.00
5	統計學	莊周	4	4	1.00
6	微積分	岳飛	4	4	1.00
7	計算機概論	陸羽	5	5	1.00

5. 請列出 1102 學期「線上課程」教學評量平均分數及總分，找出大受好評的線上課程。

5.sql - DESKTOP-6...ourse (mssql (54)) | SQLQuery24.sql -...ourse (mssql (53))*

```
CREATE OR ALTER VIEW online_course_1102_feedback AS
SELECT F.course_ID,SUM(feedback_rank) feedback_sum,
    CAST(AVG(CAST(feedback_rank AS real)) AS NUMERIC(3,2)) feedback_avg
FROM feedback F JOIN course C ON F.course_ID = C.course_ID
WHERE dbo.get_semester(C.semester) = '1102' AND C.course_is_online = '線上'
GROUP BY F.course_ID
GO
SELECT subject_name,T.teacher_name,feedback_sum,feedback_avg
FROM online_course_1102_feedback target
    JOIN course C ON target.course_ID = C.course_ID
    JOIN subject SUB ON C.subject_ID = SUB.subject_ID
    JOIN teacher T ON SUB.teacher_ID = T.teacher_ID
ORDER BY feedback_avg DESC,feedback_sum DESC
```

118 %

結果 | 訊息

	subject_name	teacher_name	feedback_sum	feedback_avg
1	計算機概論	陸羽	13	4.33
2	虛擬實境	劉邦	17	4.25
3	音樂欣賞	巴哈	33	4.13
4	微積分	岳飛	11	3.67
5	統計學	莊周	11	3.67
6	演算法	達文西	7	3.50
7	經濟學	孔丘	6	3.00

資料庫內容

dept

dept_ID	dept_name	degree
ch1	中文所	碩士
ch22	中文所	博士
lan	文學系	學士
art	表演藝術系	學士
ch12	紅學所	博士
m0	音樂系	學士
p0	哲學系	學士
p1	哲學系	碩士
p2	哲學系	博士
m11	國樂研究所	碩士
m12	國樂研究所	博士
502	資訊工程系	學士
522	資訊工程研究所	碩士
552	資訊工程研究所	博士
m21	管樂研究所	碩士
m22	管樂研究所	博士
201	數學系	學士
221	數學系	碩士
251	數學系	博士

dept_class

dept_class_ID	dept_ID	admission_year	class
107201A	201	107	A
107201B	201	107	B
107201C	201	107	C
108201A	201	108	A
108201B	201	108	B
109201A	201	109	A
109201B	201	109	B
110201A	201	110	A
110201B	201	110	B
100502A	502	100	A
107502A	502	107	A
107502B	502	107	B
108502A	502	108	A
108502B	502	108	B
109502A	502	109	A
109502B	502	109	B
110502A	502	110	A
110502B	502	110	B
107522A	522	107	A
108522A	522	108	A
109522A	522	109	A
110522A	522	110	A
art107	art	107	A
110ch1B	ch1	110	B
110ch12B	ch12	110	B
lan110B	lan	110	B
109m0	m0	109	A
109m11	m11	109	A
107p0	p0	107	A
110p1	p1	110	A

building

building_ID	building_name
A	文學一館
C2	文學二館
E	工程一館
E1	工程二館
E2	工程三館
E3	工程四館
E4	機電實驗室
E5	大型力學實驗室
E6	工程五館
H2	理學院教學館
HK	客家學院大樓
I	管理一館
I1	管理二館
IL	國鼎光電大樓
L3	國鼎圖書資料館
LS	人文社會科學大樓
M	鴻經館
O	綜教館
R2	太空及遙測研究中心
R3	研究中心大樓二期
S	科學一館
S1	科學二館
S2	科學三館
S4	科學四館
S5	科學五館
TR	教學研究綜合大樓暨大禮堂
YH	依仁堂

subject

subject_ID	subject_name	subject_credit	teacher_ID
AAAAA	A咖的演技	5	Ssn12
A0011	午休的環境	2	Ssn11
A0009	文學的模糊之美	2	Ssn9
A0017	何謂清楚--形式主義的精神	2	Ssn8
A0002	計算機概論	3	Ssn5
A0006	音樂欣賞	2	Ssn2
A0015	時間管理	3	Ssn10
A0013	時間管理	2	Ssn11
A0005	統計學	3	Ssn4
old1	程式讀卡注意事項	2	Ssn10
A0003	虛擬實境	3	Ssn7
A0010	微積分	2	Ssn10
A0001	微積分	2	Ssn3
A0004	經濟學	3	Ssn1
CE6039	資料庫系統	3	Ssn8
A0008	演算法	2	Ssn10
A0007	演算法	3	Ssn6
A0016	模糊邏輯導論	2	Ssn9
A0012	樂理導論	3	Ssn2
A0014	線上教學相關科技	2	Ssn7
CE2003	離散數學	3	Ssn8

room

room_ID	room_name	building_ID	room_limit
C21	001	C2	50
E6-A209	E6-A209	E6	50
I1-018	I1-018	I1	50
I1-304	I1-304	I1	50
K205	K205	E	50
K210	K210大教室	E	100
L102	L102	E6	50
LS1	001	LS	50
LS2	戲劇拍攝模擬場地	LS	500
O-214	O-214	O	100

score

course_ID	student_ID	student_score
100old1	100502001	71.26
100old1	100502002	71.26
100old1	nope	100.00
100old2	100502001	71.60
100old2	100502002	71.68
100old2	nope	100.00
100old3	100502001	71.06
100old3	100502002	71.66
100old3	nope	100.00
121111	star1	100.00
A00011102	110201505	98.00
A00011102	110201506	55.00
A00011102	110201507	67.00
A00021101	wrong	4.01
A00021102	110201505	49.00
A00021102	110201507	78.00
A00021102	110522002	74.00
A00031102	110201506	87.00
A00031102	110522002	80.00
A00031102	110522005	78.00
A00031102	110522006	65.00
A00041102	110201505	78.00
A00041102	110201507	89.00
A00051102	110502503	63.00
A00051102	110502504	46.00
A00051102	110522003	96.00
A00061102	110201506	62.00
A00061102	110201507	60.00
A00061102	110522002	98.00
A00061102	110522003	56.00
A00061102	110522005	55.00
A00061102	110522006	78.00
A00061102	star1	100.00
A00061102	wrong	99.99
A00071102	110522002	68.00
A00071102	110522007	87.00
A1111	aaaaaaaaa	97.89

student

student_ID	student_name	cur_dept_class_ID	cur_status
quit	孟庭修	100502A	退學
100502001	校友一	100502A	畢業
100502002	校友二	100502A	畢業
nope	卜回探	100502A	畢業
star1	chou	109m0	在學
wrong	鄒曆徐	109m0	在學
110201502	孫尚香	110201A	休學
110201503	周瑜	110201A	在學
110201504	黃蓋	110201A	在學
110201505	趙雷	110201A	在學
110201506	關興	110201A	在學
110201507	夏侯惇	110201A	在學
110201501	張飛	110201A	在學
110502502	龐統	110502A	休學
110502503	大喬	110502A	在學
110502504	甘寧	110502A	在學
110502505	司馬昭	110502A	在學
110502506	馬超	110502A	在學
110502501	關羽	110502A	在學
110522002	華陀	110522A	在學
110522003	劉備	110522A	在學
110522004	呂布	110522A	在學
110522005	諸葛亮	110522A	在學
110522006	呂蒙	110522A	在學
110522007	郭嘉	110522A	在學
110522001	華雄	110522A	退學
pstudent1	蘇格拉底	110p1	休學
aaaaaaaaa	早今天	art107	在學

feedback

course_ID	student_ID	feedback_rank
100old1	100502001	4
100old1	100502002	5
100old1	quit	1
100old2	100502001	3
100old2	100502002	3
100old2	quit	1
100old3	100502001	4
100old3	100502002	4
100old3	quit	1
121111	star1	1
A00011102	110201505	4
A00011102	110201506	5
A00011102	110201507	2
A00021101	wrong	3
A00021102	110201505	5
A00021102	110201507	3
A00031102	110201506	5
A00031102	110522002	5
A00031102	110522005	3
A00031102	110522006	4
A00041102	110201505	4
A00041102	110201507	2
A00051102	110502503	4
A00051102	110502504	5
A00051102	110522003	2
A00061102	110201506	5
A00061102	110201507	3
A00061102	110522002	4
A00061102	110522003	5
A00061102	110522005	5
A00061102	110522006	5
A00061102	star1	1
A00061102	wrong	5
A00071102	110522002	3
A00071102	110522007	4
A1111	aaaaaaaaa	5

course

course_ID	subject_ID	semester	room_ID	course_time	course_limit	course_status	course_is_online
81111	A0008	[111,1]	L102	{"二":"12"}	50	授課	並行
91111	A0009	[111,1]	C21	{"四":"56"}	50	授課	並行
101111	A0010	[111,1]	K205	{"一":"12"}	50	授課	實體
111111	A0011	[111,1]	O-214	{"一":"Z","二":"Z"}	50	授課	實體
121111	A0012	[111,1]	LS1	{"三":"56"}	10	授課	實體
131111	A0013	[111,1]	NULL	{"三":"Z","四":"Z"}	300	授課	線上
141111	A0014	[111,1]	E6-A209	{"一":"34"}	50	授課	線上
151111	A0015	[111,1]	NULL	{"三":"123"}	300	授課	線上
161111	A0016	[111,1]	NULL	{"五":"234"}	100	授課	線上
171111	A0017	[111,1]	NULL	{"五":"234"}	100	授課	線上
A1111	AAAAA	[111,1]	LS2	{"三":"BCDEF"}	1	授課	實體
CE20031111	CE2003	[111,1]	L102	{"四":"56","五":"56"}	50	授課	並行
100old1	old1	[100,1]	O-214	{"四":"12"}	50	歷史資料	實體
100old2	A0002	[100,2]	L102	{"一":"567"}	50	歷史資料	實體
100old3	A0007	[100,2]	E6-A209	{"二":"567"}	50	歷史資料	實體
A00021101	A0002	[110,1]	E6-A209	{"五":"123"}	50	歷史資料	實體
81101	A0008	[110,1]	E6-A209	{"一":"12"}	50	歷史資料	並行
A00011102	A0001	[110,2]	K210	{"一":"567"}	50	歷史資料	線上
A00021102	A0002	[110,2]	L102	{"二":"34","五":"4"}	50	歷史資料	線上
A00031102	A0003	[110,2]	E6-A209	{"四":"567"}	50	歷史資料	線上
A00041102	A0004	[110,2]	I1-018	{"四":"567"}	50	歷史資料	線上
A00051102	A0005	[110,2]	I1-304	{"五":"234"}	50	歷史資料	線上
A00061102	A0006	[110,2]	O-214	{"三":"56"}	100	歷史資料	線上
A00071102	A0007	[110,2]	L102	{"三":"234"}	50	歷史資料	線上
81102	A0008	[110,2]	L102	{"五":"12"}	50	歷史資料	實體
141102	A0014	[110,2]	E6-A209	{"四":"34"}	50	歷史資料	線上
CE60391111	CE6039	[110,2]	L102	{"一":"567"}	50	歷史資料	並行

teacher

teacher_ID	teacher_name
Ssn1	孔丘
Ssn10	歐德曹
Ssn11	卜伍修
Ssn12	晚今天
Ssn2	巴哈
Ssn3	岳飛
Ssn4	莊周
Ssn5	陸羽
Ssn6	達文西
Ssn7	劉邦
Ssn8	禁孟峰
Ssn9	禁孟峰

selection

course_ID	student_ID	select_dept_class_ID	select_type	select_result
100old1	100502001	100502A	必修	中選
100old1	100502002	100502A	必修	中選
100old1	nope	100502A	必修	中選
100old1	quit	100502A	必修	中選
100old2	100502001	100502A	必修	中選
100old2	100502002	100502A	必修	中選
100old2	nope	100502A	必修	中選
100old2	quit	100502A	必修	中選
100old3	100502001	100502A	必修	中選
100old3	100502002	100502A	必修	中選
100old3	nope	100502A	必修	中選
100old3	quit	100502A	必修	中選
121111	star1	109m0	必修	中選
A00011102	110201502	110201A	必修	中選
A00011102	110201505	110201A	必修	中選
A00011102	110201506	110201A	必修	中選
A00011102	110201507	110201A	必修	中選
A00021101	wrong	109502B	必修	中選
A00021102	110201505	110201A	必修	中選
A00021102	110201507	110201A	必修	中選
A00021102	110502502	110502A	必修	中選
A00021102	110522001	110522A	必修	中選
A00021102	110522002	110522A	必修	中選
A00031102	110201506	110201A	選修	中選
A00031102	110502502	110502A	選修	中選
A00031102	110522001	110522A	選修	中選
A00031102	110522002	110522A	選修	中選
A00031102	110522005	110522A	選修	中選
A00031102	110522006	110522A	選修	中選
A00031102	110522003	110522A	選修	落選
A00041102	110201502	110201A	必修	中選
A00041102	110201505	110201A	必修	中選
A00041102	110201507	110201A	必修	中選
A00041102	110502502	110502A	必修	落選
A00051102	110201502	110201A	選修	中選
A00051102	110502503	110502A	選修	中選
A00051102	110502504	110502A	選修	中選
A00051102	110522003	110522A	選修	中選
A00061102	star1	109m0	必修	中選
A00061102	wrong	109m0	必修	中選
A00061102	110201506	110201A	選修	中選
A00061102	110201507	110201A	選修	中選
A00061102	110502502	110502A	選修	中選
A00061102	110522001	110522A	選修	中選
A00061102	110522002	110522A	選修	中選
A00061102	110522003	110522A	選修	中選
A00061102	110522005	110522A	選修	中選
A00061102	110522006	110522A	選修	中選
A00061102	110201502	110201A	選修	落選
A00061102	110502505	110502A	選修	落選
A00061102	110502506	110502A	選修	落選
A00071102	110522001	110522A	選修	中選
A00071102	110522002	110522A	選修	中選
A00071102	110522007	110522A	選修	中選
A1111	aaaaaaaaa	art107	選修	中選