**Team**

Carlos Parlour

**Abstract**

This database will

**Mission Statement**

**Mission Objectives**

To maintain data on campuses

To maintain data on buildings

To maintain data on departments

To maintain data on Major declaration

To maintain data on staff

To maintain data on courses

To maintain data on students

Perform searches on campuses

Perform searches on buildings

Perform searches on departments

Perform searches on staff

Perform searches on courses

Perform searches on students

Track the status of classes

Track the status of rating

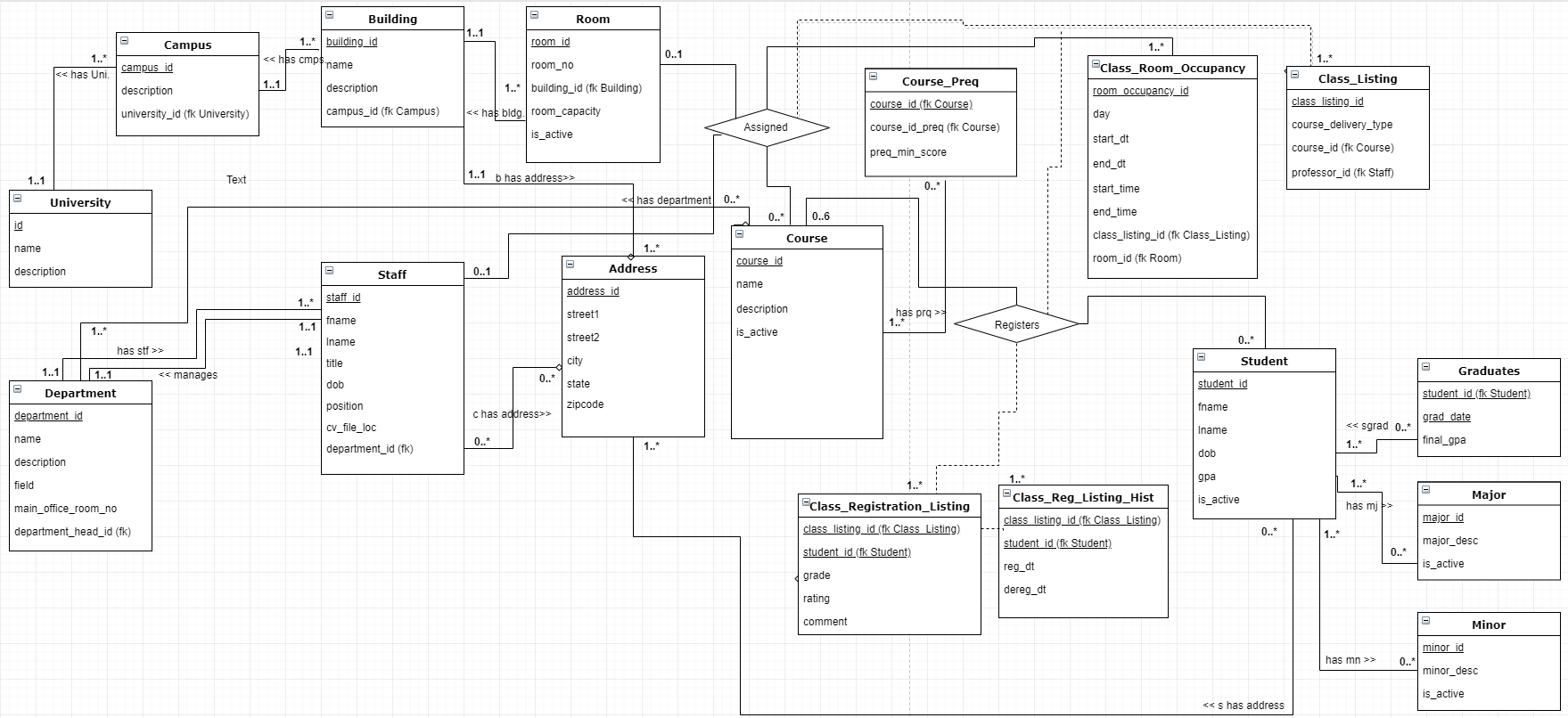
Report on classes

Report on staff

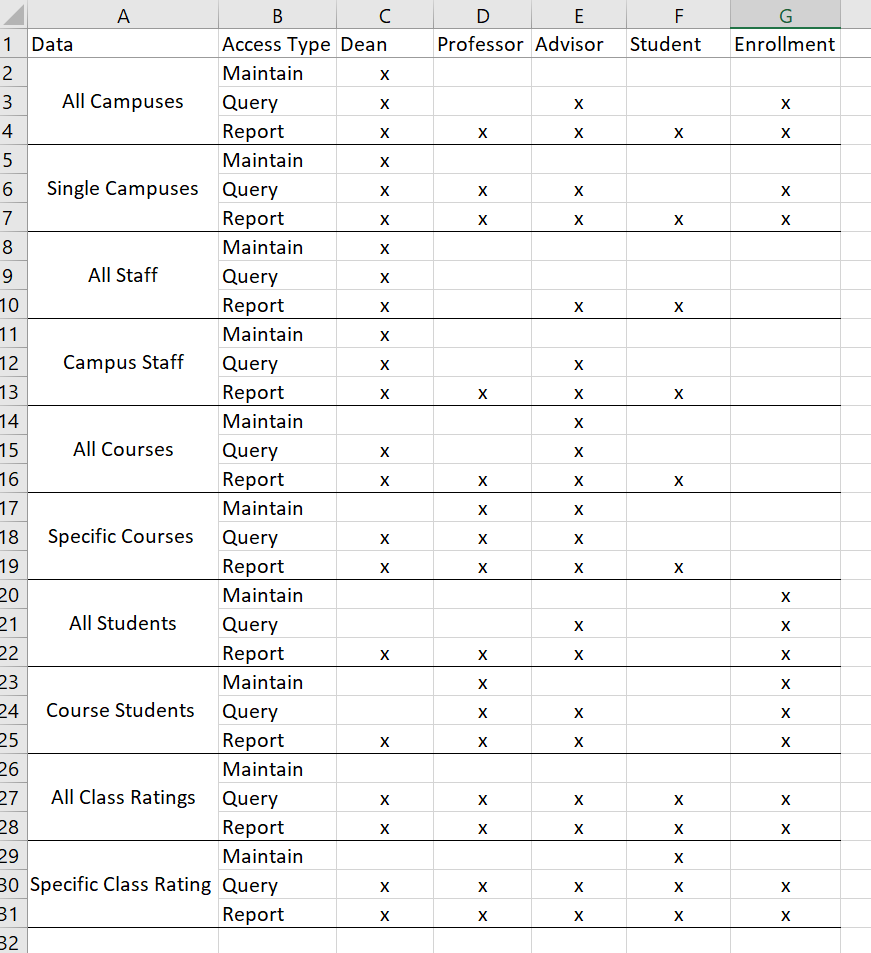
Report on courses

Report on students

**ER Model**

****

**Major User Views**



**Use Cases**

**Case name:** Anyone searches classes

Actor: Student, Dean, Professor, Enrollment Officer

Steps:

1. User clicks “Search Classes” button
2. Parameters for search are entered
3. Results appear

**SQL Statements**

SELECT \*

FROM Class\_Listing cl

JOIN Course c on c.course\_id = cl.course\_id

left outer JOIN Class\_Room\_Occupancy cro on cro.class\_listing\_id = cl.class\_listing\_id

left outer JOIN Room r on r.room\_id = cro.room\_id

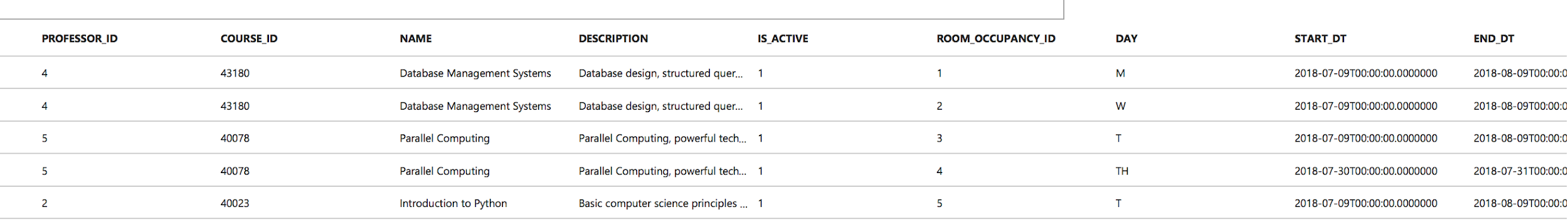
JOIN Staff s on s.staff\_id = cl.professor\_id

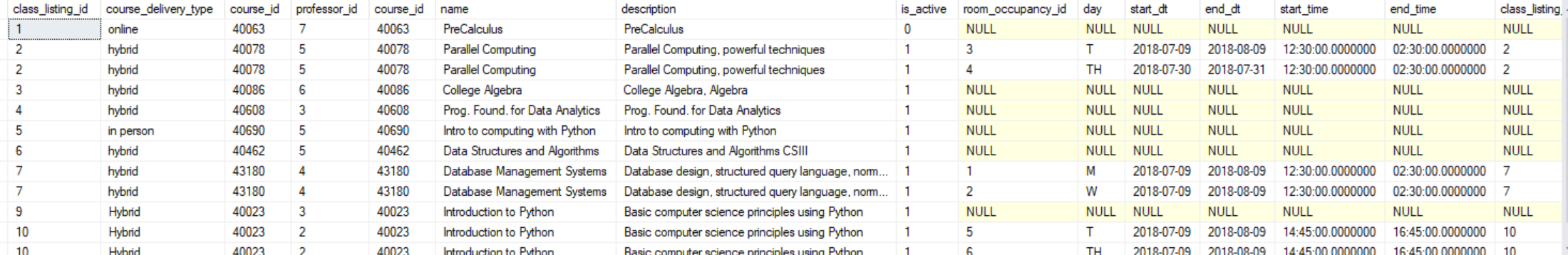
left outer JOIN Building b on b.building\_id = r.building\_id

left outer JOIN Building\_Addr ba on ba.building\_id = b.building\_id

left outer JOIN Campus cmp on cmp.campus\_id = b.campus\_id

Expected: Displays all courses, courses for computer science department, courses as otherwise specified by search terms

Actual: not all class listings were displayed, edited query above



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student compares class ratings

Actor: Student

Steps:

1. User searches classes and results displayed
2. User selects results to compare
3. “Compare Courses” button clicked
4. Sample of weekly schedule is displayed, showing class times and locations
5. Course/Instructor reviews are displayed in side panel, grouped by instructor

**SQL Statements**

SELECT \*

FROM Class\_Listing cl

JOIN Course c on c.course\_id = cl.course\_id

left outer Join Course\_Dept cd on cd.course\_id = c.course\_id

left outer JOIN Class\_Room\_Occupancy cro on cro.class\_listing\_id = cl.class\_listing\_id

left outer JOIN (SELECT cl.course\_id, professor\_id, sum(rating)/count(rating) overall\_course\_rating

FROM Class\_Registration\_Listing crl JOIN Class\_Listing cl on cl.class\_listing\_id = crl.class\_listing\_id

GROUP BY cl.course\_id, cl.professor\_id) a on a.course\_id = cl.course\_id and a.professor\_id = cl.professor\_id

left outer JOIN Room r on r.room\_id = cro.room\_id

left outer JOIN Staff s on s.staff\_id = cl.professor\_id

left outer join Building b on b.building\_id = r.building\_id

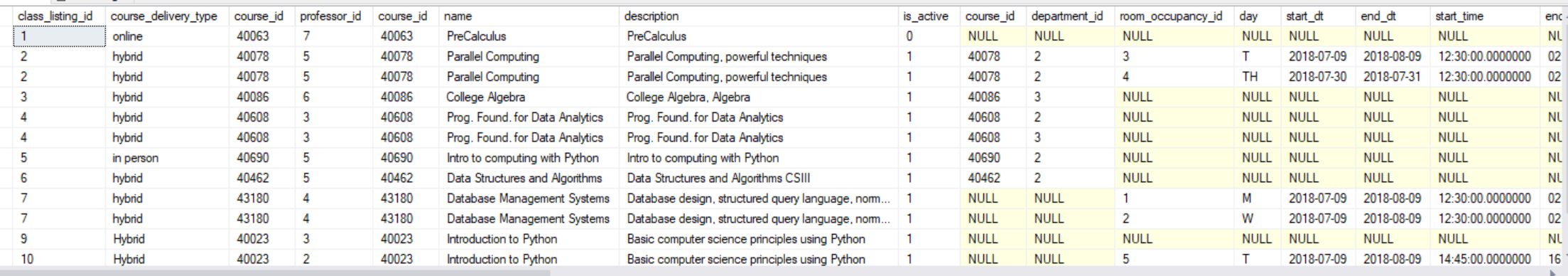
left outer JOIN Building\_Addr ba on ba.building\_id = b.building\_id

left outer JOIN Campus cmp on cmp.campus\_id = b.campus\_id

WHERE ; -- where dynamically updated parameters match query

Expected: display all available information about courses offered along with the overall rating for the course and professor matching the class\_listings

Actual: not all class\_listings displayed, edited query above



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student adds course for registration

Actor: Student

Steps:

1. User searches classes and results displayed
2. User selects course
3. “Add to Schedule” button is clicked
4. Course is added to registration ‘cart’
5. Registration is completed by clicking “Finalize Registration” button on registration page

**SQL Statements**

INSERT into Class\_Registration\_Listing (class\_listing\_id, student\_id)

values (4,4);

Expected:new row added to table

Actual: as expected



On Insert Trigger

CREATE TRIGGER insClassRegList

ON Class\_Registration\_Listing

FOR INSERT

AS

INSERT INTO Class\_Reg\_Listing\_Hist (class\_listing\_id, student\_id, reg\_dt)

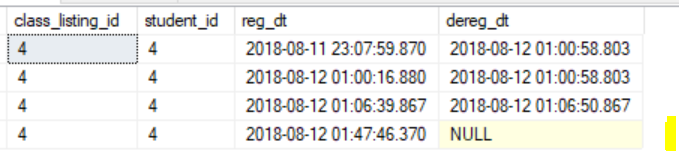
SELECT class\_listing\_id, student\_id, CURRENT\_TIMESTAMP

FROM Inserted

GO;

Expected: inserted value from class\_registration\_listing is inserted into class-reg\_listing\_hist with a timestamp of the insertion time

Actual:as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student removes class from schedule

Actor: Student

Steps:

1. User clicks “Remove Classes” button
2. Student selects class
3. Student presses “Remove from schedule”

**SQL Statements**

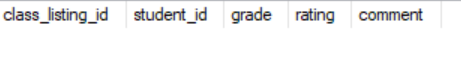
DELETE FROM Class\_Registration\_Listing

WHERE class\_listing\_id = 4 and student\_id = 4

Expected: row deleted

Actual: as expected

SELECT \* FROM class\_registration\_listing where class\_listing\_id = 4 and student\_id = 4



On Delete Trigger:

CREATE TRIGGER delClassRegList

ON Class\_Registration\_Listing

AFTER Delete

AS

BEGIN

UPDATE crh

set dereg\_dt = CURRENT\_TIMESTAMP

FROM Class\_Reg\_Hist\_Listing crh

JOIN DELETED d on crh.class\_listing\_id = d.class\_listing\_id and crh.student\_id = d.student\_id

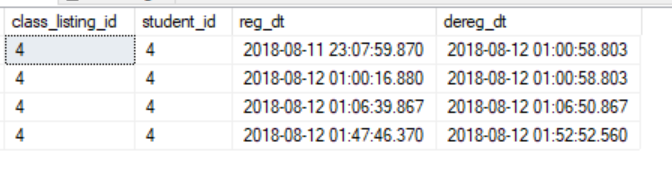
where crh.reg\_dt is not null and crh.dereg\_dt is null

END

GO;

Expected: dereg\_dt is updated to show the current time when the update occurred

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student updates address

Actor: Student

Steps:

1. Student clicks Update Address button
2. Current address information is displayed
3. Student makes changes and clicks save

**SQL Statements**

UPDATE ad

SET street1 = '124 hollywood', street2 = null, city = 'austin', state = 'texas', zipcode = 77012

FROM Address ad

JOIN Student\_Addr sa on sa.address\_id = ad.address\_id

WHERE sa.student\_id = 1

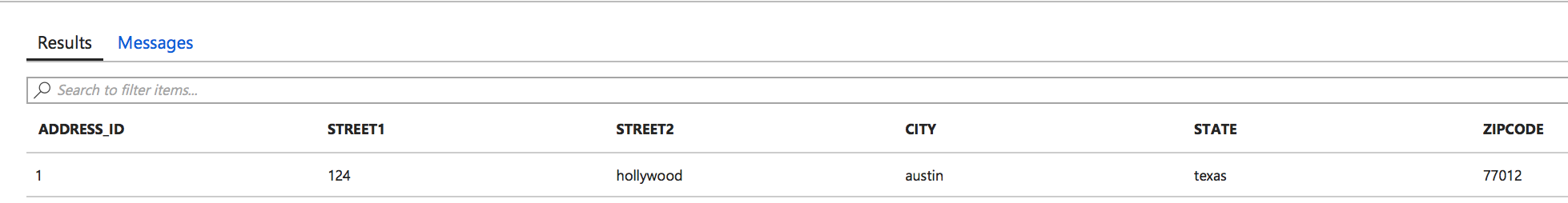
select address\_id, street1, street2, city, state, zipcode

from Address

where address\_id = 1

Expected: student address updated

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student review classes registered

Actor: Student

Steps:

1. Student clicks Review Class Registration
2. Class registration information displayed

**SQL Statements**

SELECT a.\*, crl.\*

FROM Class\_Registration\_Listing crl

JOIN (

SELECT student\_id,count(class\_listing\_id) num\_classes\_reg\_total

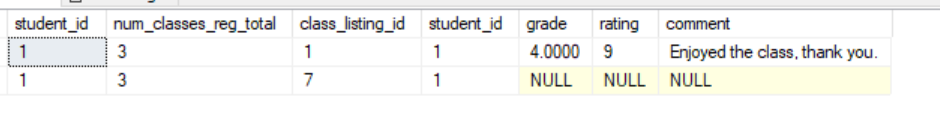
FROM Class\_Reg\_hist\_Listing

WHERE student\_id = 1 and reg\_dt between '2018-07-01' and '2018-10-01' and dereg\_dt is null

GROUP BY student\_id) a on a.student\_id = crl.student\_id

Expected: displayed class registration information with grade

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Enrollment officer searches for student

Actor: Enrollment Officer

Steps:

1. User clicks Search Student button
2. Parameters for search are entered, options first name last name, dob, course history
3. Enrollment officer clicks search
4. Results are displayed

**SQL Statements**

SELECT \*

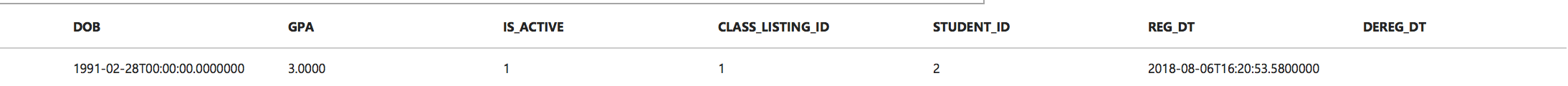
FROM Student s

JOIN Class\_Reg\_Hist\_Listing crh on crh.student\_id = s.student\_id

WHERE s.fname = ‘carlos’

Expected: displayed registered courses for students with first name Carlos

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Enrollment reviews number of active students

Actor: Enrollment Officer

Steps:

1. User clicks Review Student Statistics
2. number of active students displayed

**SQL Statements**

SELECT count(is\_active) num\_active\_students

FROM Student

WHERE is\_active = 1

Group by is\_active

Expected: number of active students displayed

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** New Student is enrolled

Actor: Enrollment Officer

Steps:

1. User clicks “New Student” button
2. New student id is generated and displayed, fields Major, Minor, Graduation\_Date, and GPA are null
3. Student information is entered for required fields
4. All information is displayed for confirmation
5. “Complete Enrollment” button is clicked

**SQL Statements**

INSERT INTO Student

VALUES (17,'jennifer','Boake','1989-01-01',1);

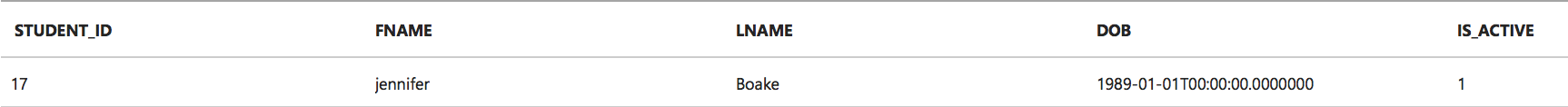
select student\_id, fname, lname,dob, is\_active

from Student

where fname = 'jennifer'

Expected: new tuple added to table

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student dob or name is edited

Actor: Enrollment Officer

Steps:

1. Officer searches student
2. Enrollment officer clicks “Edit”
3. Fields for first name, last name, and date of birth are editable.
4. Officer makes edits and clicks Save
5. Information is displayed for review, officer clicks Confirm Changes button

**SQL Statements**

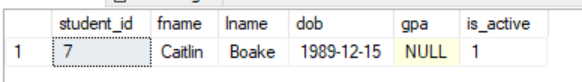
Update Student

set DOB = '1989-12-15'

where student\_id = 7

Expected: date of birth field is updated for row with student\_id 7

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Duplicate Student is deleted

Actor: Enrollment Officer

Steps:

1. Officer searches student
2. Enrollment officer clicks “List as Duplicate” button
3. Window appears with warning, officer clicks “confirm delete” button
4. Student entry is scheduled to be deleted from database

**SQL Statements**

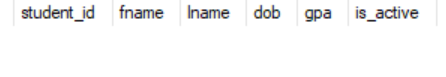
DELETE FROM Student

WHERE student\_id = 7

(1 row affected)

Expected: row with student\_id 7 is removed from table

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student adds rating to class

Actor: Student

Steps:

1. Student selects “Display Class Schedule”
2. Student select submit feedback next to each course.
3. Student selects save once completed.

**SQL Statements**

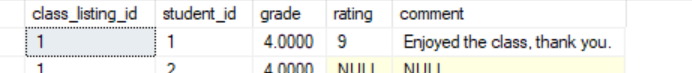
UPDATE Class\_Registration\_Listing

SET rating = 9, comment = 'Enjoyed the class, thank you.'

WHERE student\_id = 1 and class\_listing\_id = 1

Expected: fields rating and comment are updated with data

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student declares major

Actor: Advisor

Steps:

1. User searches and locates Student
2. Advisor clicks Edit Student button
3. Student’s major is entered in to Major field
4. All Information displayed for confirmation
5. “Save Changes” button is clicked

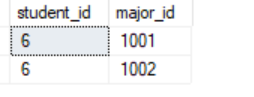
**SQL Statements**

INSERT INTO Student\_Major

VALUES (6,1002)

Expected: new row is added to student\_major table

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Advisor adds Course to database

Actor: Advisor

Steps:

1. Faculty member clicks “Add new course”
2. Course\_ID field is automatically populated
3. Enters the name of the course
4. Selects department
5. “Save Changes” is clicked

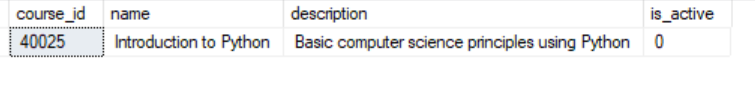
**SQL Statements**

INSERT INTO Course

VALUES (40025,'Introduction to Python','Basic computer science principles using Python',0)

Expected: new row in course is added

Actual: as expected



INSERT INTO Course\_Dept

VALUES (40608,3)

Expected: new row is added to course\_dept when new course is added

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Advisor removes course from being offered

Actor: Advisor

Steps:

1. Faculty member clicks “Search Classes”
2. Enters the name of the course
3. Faculty changes the Inactive field from False to True
4. “Save Changes” is clicked

**SQL Statements**

UPDATE Course

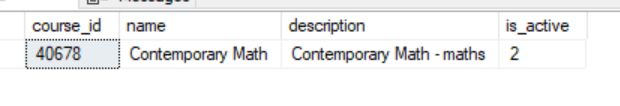
SET is\_active = 2

WHERE course\_id = 40678

-- a value of 1 for is\_active= true, 2 = false

Expected: course is\_active field is updated to 2

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Advisor searches class listings

Actor: Advisor

Steps:

1. Faculty member clicks “Search Classes”
2. Enters parameters for search and clicks Search
3. Results appear

**SQL Statements**

SELECT \*

FROM Class\_Listing cl

JOIN Course c on c.course\_id = cl.course\_id

left outer JOIN Class\_Room\_Occupancy cro on cro.class\_listing\_id = cl.class\_listing\_id

left outer JOIN Room r on r.room\_id = cro.room\_id

JOIN Staff s on s.staff\_id = cl.professor\_id

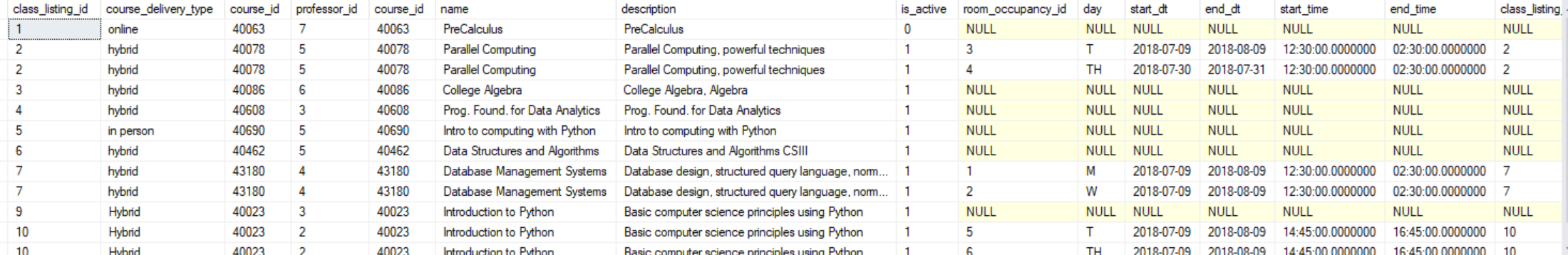
left outer JOIN Building b on b.building\_id = r.building\_id

left outer JOIN Building\_Addr ba on ba.building\_id = b.building\_id

left outer JOIN Campus cmp on cmp.campus\_id = b.campus\_id

Expected: Displays all courses, courses for computer science department, courses as otherwise specified by search terms

Actual: not all class listings were displayed, edited query above



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Advisor updates class time

Actor: Advisor

Steps:

1. Faculty member searches and locates specific class listing
2. Advisor selects edit
3. Advisor makes changes to time and clicks Save button
4. Advisor clicks Confirm Changes button

**SQL Statements**

UPDATE Class\_Room\_Occupancy

SET room\_id = 10 WHERE room\_occupancy\_id = 6

Expected: class time is updated in room\_occupancy for specific room occ. id  
Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Advisor removes student’s class from schedule

Actor: Advisor

Steps:

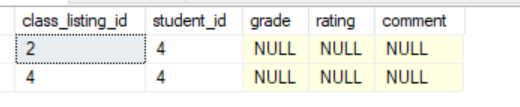
1. Student is searched and located
2. Advisor clicks Show Classes button
3. Clicks Edit Classes button
4. Selects classes to be removed from schedule
5. Clicks “remove from class schedule” button
6. Confirm, Save

**SQL Statements**

DELETE FROM Class\_Registration\_Listing

WHERE class\_listing\_id = 4 and student\_id = 4

Expected: specific row is removed from table  
Actual: as expected

before:

after: 

On Delete Trigger:

CREATE TRIGGER delClassRegList

ON dbo.Class\_Registration\_Listing

FOR Delete

AS

UPDATE crh

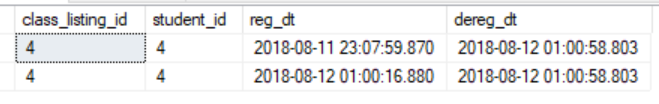
set dereg\_dt = CURRENT\_TIMESTAMP

FROM Class\_Reg\_Listing\_Hist crh

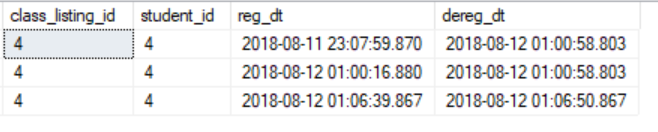
JOIN DELETED d on crh.class\_listing\_id = d.class\_listing\_id and crh.student\_id = d.student\_id

where crh.reg\_dt is not null and crh.dereg\_dt is null

GO;

Expected: dereg\_dt is updated for specific instance of student registering for class\_listing  
Actual: not expected, dereg\_dt is changed for all instances of that class\_listing\_id and student\_id combo. Altered Trigger delClassRegList on Class\_Registration\_Listing table, modification is underlined above.

after edit:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Professor searches rating

Actor: Professor

Steps:

1. Professor selects “Overall rating”.
2. Classes are displayed with his or her overall rating and comments

**SQL Statements**

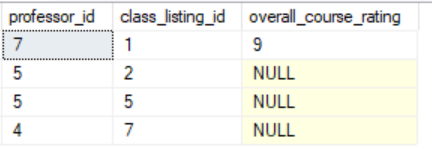
SELECT professor\_id, crl.class\_listing\_id, sum(rating) overall\_course\_rating

FROM Class\_Registration\_Listing crl

JOIN Class\_Listing cl on cl.class\_listing\_id = crl.class\_listing\_id

GROUP BY professor\_id, crl.class\_listing\_id

Expected: Overall rating for class\_listings for professor is displayed  
Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student reviews class grade

Actor: Student

Steps:

1. Student selects “Display Class Schedule”
2. Student selects “Grade” button for specific class
3. Grade for that class is displayed

**SQL Statements**

SELECT grade, class\_listing\_id

FROM Class\_Registration\_Listing

WHERE student\_id = 2 and class\_listing\_id = 7

Expected: grade displayed for specific class  
Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Student reviews grades for all classes taken

Actor: Student

Steps:

1. Student selects “Grades” button
2. Grades for all classes that student has taken are displayed

**SQL Statements**

SELECT c.name class\_name, grade

FROM Class\_Registration\_Listing reg

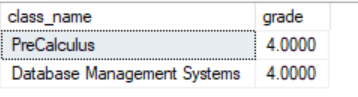
JOIN Class\_Listing cl on cl.class\_listing\_id = reg.class\_listing\_id

JOIN Course c on c.course\_id = cl.course\_id

WHERE student\_id = 2

Expected: class name and class grades are displayed

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Course is offered as Class for semester

Actor: Advisor

Steps:

1. Advisor selects “add class”
2. Advisor selects Course from a list and clicks “list this course”
3. Values class\_listing\_id and course\_id are pre-populated in form
4. Advisor selects room from list
5. Advisor selects professor from list
6. Advisor selects course delivery type from list
7. Advisor enters class dates, and class times

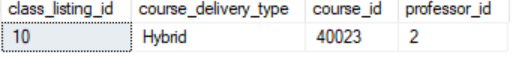
**SQL Statements**

INSERT INTO Class\_Listing

VALUES (10,'Hybrid', 40023,2)

Expected: new row is added to class\_listing

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Professor updates student grades

Actor: Professor

Steps:

1. Professor clicks Update Grades button
2. A list of classes the professor is teaching is displayed, professor clicks appropriate class
3. A list of students enrolled in this class is displayed
4. Next to each student are columns listing each class assignment
5. Professor enters a grade for a student underneath the appropriate assignment heading
6. Cumulative grade is calculated and updated automatically after professor clicks out of cell
7. Professor clicks Update button

**SQL Statements**

UPDATE Class\_Registration\_Listing

SET grade = 3.5

WHERE class\_listing\_id = 7 and student\_id = 2

Expected: grade field is updated for student\_id 2 with class\_listing\_id 7

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Professor adds new address

Actor: Professor

Steps:

1. Professor click Add address
2. Enters information
3. Clicks save

**SQL Statements**

INSERT INTO Address

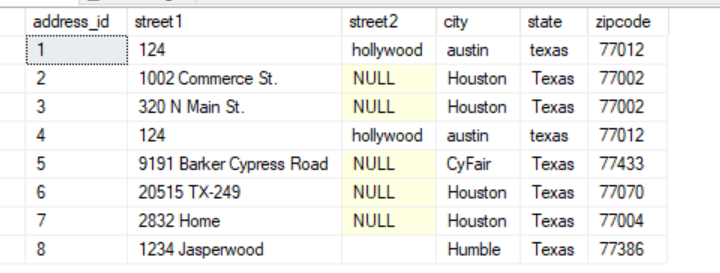
values(8, '1234 Jasperwood', 'Humble', 'tx', '77386');

INSERT INTO Staff\_Addr

VALUES (1,8)

Expected: new rows added to tables Address and Staff\_Addr

Actual: as expected





\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Professor removes student’s class from schedule

Actor: Professor

Steps:

1. Student is searched and located
2. Professor clicks Show Classes button
3. Clicks Edit Classes button
4. Selects classes to be removed from schedule
5. Clicks “remove from class schedule” button
6. Confirm, Save

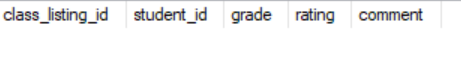
**SQL Statements**

DELETE FROM Class\_Registration\_Listing  
WHERE class\_listing\_id = vClassListingId and student\_id = vStudentId;

Expected: row deleted

Actual: as expected

SELECT \* FROM class\_registration\_listing where class\_listing\_id = 4 and student\_id = 4



On Delete Trigger:

CREATE TRIGGER delClassRegList

ON Class\_Registration\_Listing

AFTER Delete

AS

BEGIN

UPDATE crh

set dereg\_dt = CURRENT\_TIMESTAMP

FROM Class\_Reg\_Hist\_Listing crh

JOIN DELETED d on crh.class\_listing\_id = d.class\_listing\_id and crh.student\_id = d.student\_id

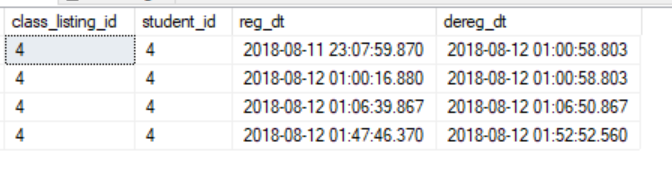
where crh.reg\_dt is not null and crh.dereg\_dt is null

END

GO;

Expected: dereg\_dt is updated to show the current time when the update occurred

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Dean searches for staff member

Actor: Dean

Steps:

1. Dean clicks “search staff”
2. Parameters like first and last name, classes taught, start date, department, and position can be entered
3. Dean clicks “search”
4. Results are displayed

**SQL Statements**

SELECT \*

FROM Staff s

left outer JOIN Class\_Listing cl on cl.professor\_id = s.staff\_id

WHERE -- where dynamically updated parameters match

Expected: all staff are available for search, some are displayed based on search parameters

Actual: not all staff are displayed, edited query above



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Dean adds new Professor to database

Actor: Dean

Steps:

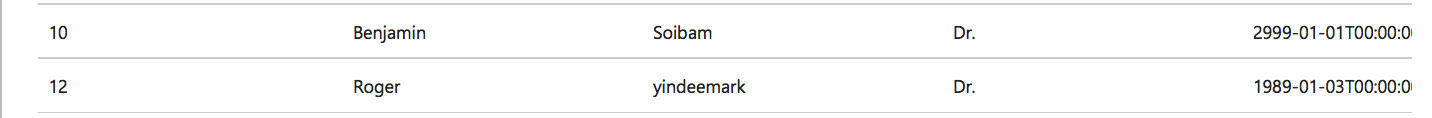
1. Dean clicks New Staff button
2. Staff\_id is pre-populated in the form
3. Department and position are selected from lists
4. Professor’s CV is attached by selecting it from a list of already uploaded CVs
5. Dean clicks Save button and confirms information is correct, clicks Confirm button

**SQL Statements**

INSERT INTO Staff

VALUES (12,'Roger','yindeemark','Dr.','1989-01-03','Associate',null,1,1);

Expected: new row inserted into Staff table  
Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Dean deletes a duplicate Staff entry

Actor: Dean

Steps:

1. Dean searches for Staff and locates duplicate entry
2. Dean clicks “List as Duplicate” button
3. Window appears with warning, dean clicks “confirm delete” button
4. Staff entry is scheduled to be deleted from database

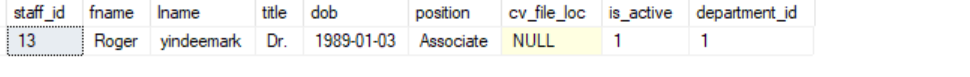
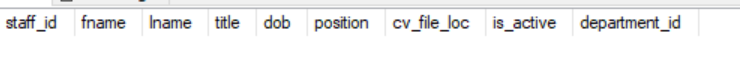
**SQL Statements**

DELETE FROM Staff

where Staff\_id = 13

Expected: staff member marked as duplicate is removed from Staff table

Actual: as expected

before: after: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Dean reviews number of active staff

Actor: Dean

Steps:

1. User clicks Review Staff Statistics
2. number of active staff displayed

**SQL Statements**

SELECT count(is\_active) num\_active\_staff

FROM Staff

WHERE is\_active = 1

Group by is\_active

Expected: number of active staff are displayed

Actual: as expected



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Case name:** Dean updates rooms to building after renovation

Actor: Dean

Steps:

1. Dean clicks “Infrastructure”
2. List of Campuses is displayed, dean selects a campus
3. List of Buildings is displayed, dean selects building
4. List of rooms is displayed
5. Click Add Room button
6. Room information is entered, room\_id and building\_id are pre-populated in form
7. Dean clicks Save button

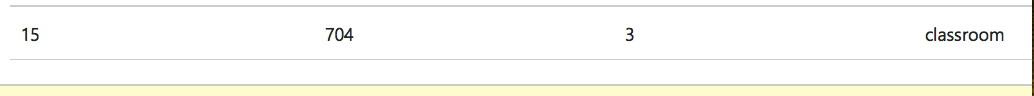
**SQL Statements**

INSERT INTO Room

VALUES (15,704,3,'classroom',50,0);

Expected: new row inserted into table Room

Actual: as expected



**Relational Model**

University(university\_id, name, description)

Address(address\_id, street1, street2, city, state, zipcode)

Campus (campus\_id, description, university\_id (fk University))

Campus\_Addr (campus\_id (fk Campus), address\_id (fk Address))

Building (building\_id, name, description, campus\_id (fk Campus))

Building\_Addr (building\_id (fk Building ), address\_id (fk Address))

Room (room\_id, room\_number, building\_id (fk Building), room\_capacity, is\_active)

Department (department\_id, name, description, room\_no (fk Room), department\_head\_id (fk Staff))

Staff (staff\_id, fname, lname, title, dob, position, cv\_file\_loc, department\_id (fk Department), is\_active)

Staff\_Addr(staff\_id (fk Staff), address\_id (fk address))

Campus\_Addr (campus\_id (fk Campus), address\_id (fk Address))

Course (course\_id, name, description, is\_active)

Course\_Dept(course\_id (fk Course), department\_id (fk Department))

Course\_Preq (course\_id (fk Course), course\_id\_preq(fk Course), preq\_min\_score)

Class\_Listing (class\_listing\_id, course\_delivery\_type, course\_id (fk Course), professor\_id (fk Staff))

Class\_Room\_Occupancy (room\_occupancy\_id, day, start\_dt, end\_dt, start\_time, end\_time,class\_listing\_id (fk Class\_Listing), room\_id (fk Room))

Student (student\_Id, fname, lname, dob, GPA, is\_active)

Student\_Minor(student\_id (fk Student), minor\_id (fk Minor))

Minor(minor\_id, minor\_desc, is\_active)

Major(major\_id, major\_desc, is\_active)

Student\_Major(student\_id (fk Student), major\_id (fk Major))

Student\_Addr(student\_id(fk Student), address\_id (fk Address))

Graduates (student\_id (fk Student), grad\_date, final\_GPA)

Class\_Registration\_Listing (class\_listing\_id (fk Class\_Listing) , student\_id (fk Student), grade, rating, comment)

Class\_Reg\_Listing\_Hist (reg\_hist\_id, class\_listing\_id (fk Class\_Listing), student\_id (fk Student), reg\_dt, dereg\_dt)

**Functional Dependencies:**

university\_id -> name, description

address\_id -> street1, street2, city, state, zipcode

campus\_id -> name, description, university\_id

building\_id -> name, description, campus\_id

room\_id -> room\_number, building\_id, room\_capaticy, inactive

department\_id -> name, description, room\_id, department\_head\_id

staff\_id -> fname, lname, title, dob, department\_id, position, cv\_file\_loc, is\_active

course\_id -> name, description, is\_active

course\_id, course\_id\_preq -> preq\_min\_score

class\_listing\_id -> course\_id, professor\_id, course\_delivery\_type

room\_occupancy\_id -> day, start\_dt, end\_dt, start\_time, end\_time, class\_listing\_id, room\_id

student\_id -> fname, lname, is\_active, dob, GPA

student\_id, grad\_dt -> final\_gpa

major\_id -> major\_desc, is\_active

minor\_id -> minor\_desc, is\_active

class\_listing\_id, student\_id -> grade, rating, comment

class\_listing\_id, student\_id, date\_registered -> date\_deregistered

All relations are in 4NF, see BCNF verification at end of document.

**SQL Statements**

These statements below will show one insert, one delete, one update, one aggregate, one joint query query for each entity and its direct relationship if applicable.

----------------------------------------

---- University

----------------------------------------

SELECT \* FROM university

--INSERT INTO University

--VALUES (2,'University of Houston','University of Houston')

--UPDATE University

--SET Name = 'University of Houston Downtown'

--WHERE university\_Id = 1

--DELETE FROM University

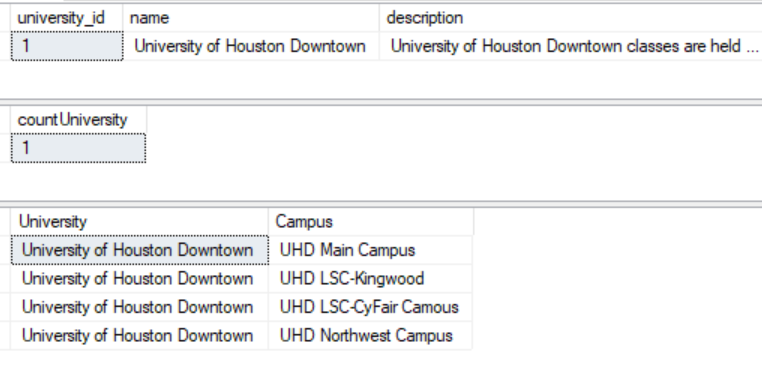
--WHERE University\_Id = 2

SELECT count(\*) countUniversity FROM University

SELECT u.name as University, c.description as Campus

FROM University u

join Campus c on c.university\_id = u.University\_Id



----------------------------------------

---- Address

----------------------------------------

SELECT \* FROM address

--INSERT INTO Address

--VALUES (8, '1234 Home Address', 'Apt. 102', 'Houston', 'Texas','77012')

--UPDATE Address

--SET zipcode = 77009

--WHERE address\_id = 8

--DELETE FROM Address

--WHERE address\_id = 8

SELECT count(\*) houston\_addresses

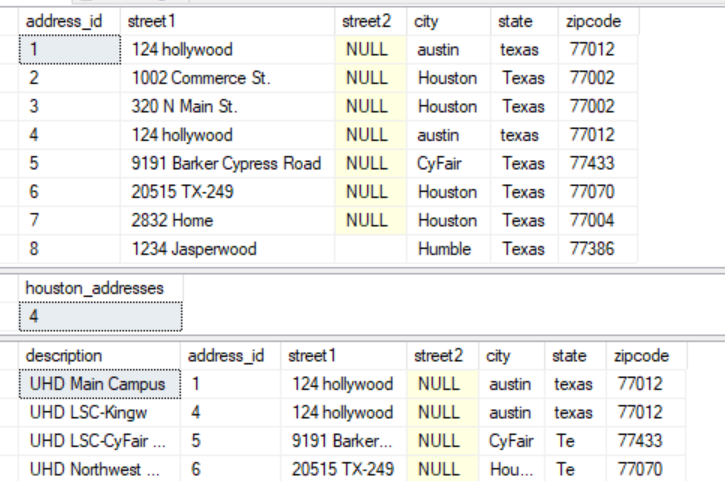
FROM address

WHERE city = 'Houston'

SELECT c.description, a.\* FROM Address a

join Campus\_Addr ca on ca.address\_id = a.address\_id

join Campus c on c.campus\_id = ca.campus\_id



----------------------------------------

---- Campus

----------------------------------------

SELECT \* FROM campus

--INSERT INTO Campus

--VALUES (5,'University of Houston - Main Campus',1)

--UPDATE Campus

--SET university\_id = 2 WHERE campus\_id = 5

--DELETE FROM Campus

--WHERE campus\_id = 5

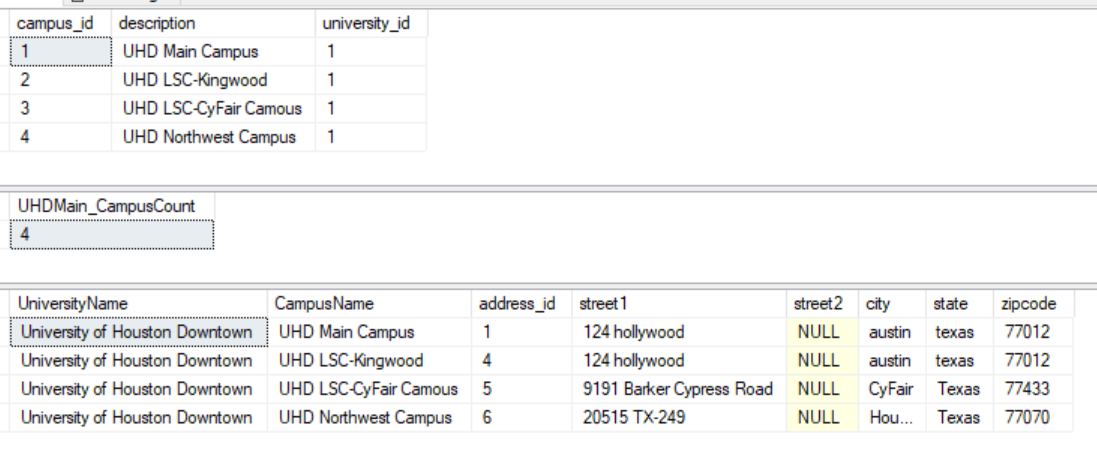
SELECT count(campus\_id) UHDMain\_CampusCount FROM campus WHERE university\_id = 1

SELECT u.Name UniversityName, c.description CampusName, a.\* FROM campus c

join university u on c.university\_id = u.university\_id

join Campus\_Addr ca on ca.campus\_id = c.campus\_id

join Address a on a.address\_id = ca.address\_id



----------------------------------------

---- Campus\_Addr

----------------------------------------

SELECT \* FROM Campus\_Addr

--INSERT INTO Campus\_Addr

--VALUES (1,4)

--UPDATE Campus\_Addr

--SET address\_id = 1

--WHERE campus\_id = 1

--DELETE FROM Campus\_Addr

--WHERE campus\_id = 1 and address\_id = 4

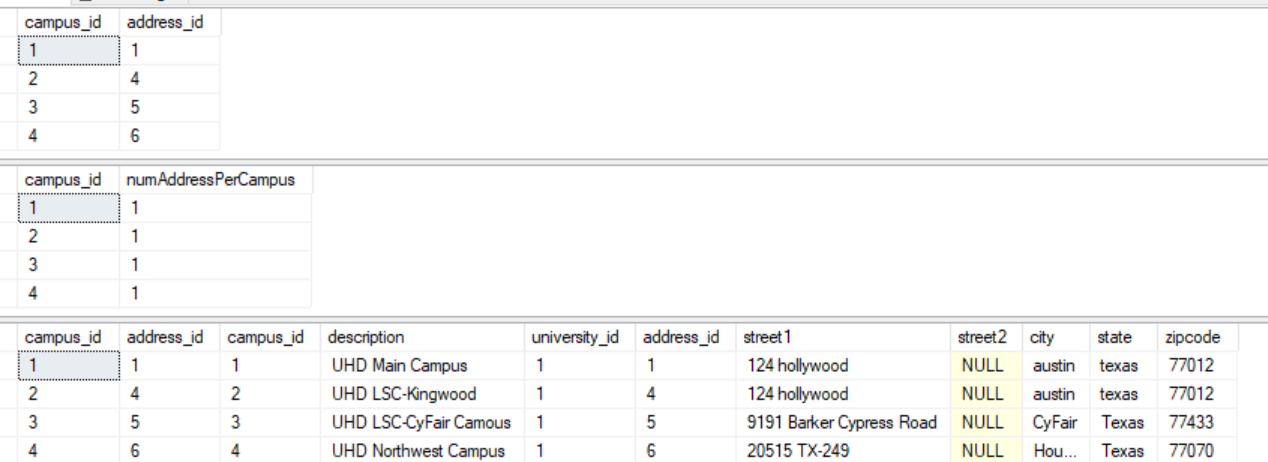
SELECT campus\_id, count(address\_id) numAddressPerCampus FROM Campus\_Addr

GROUP BY campus\_id

SELECT \* FROM Campus\_Addr ca

join Campus c on ca.campus\_id = c.campus\_id

join Address a on a.address\_id = ca.address\_id



----------------------------------------

---- Building

----------------------------------------

SELECT \* FROM building

--INSERT INTO Building

--VALUES (4,'Building 11', 'Corporate Office',4)

--UPDATE Building

--SET description = 'Corporate Office Northwest'

--WHERE building\_id = 4

--DELETE FROM Building

--WHERE building\_id = 4

SELECT campus\_id, count(building\_id) numBuildings

FROM BUILDING

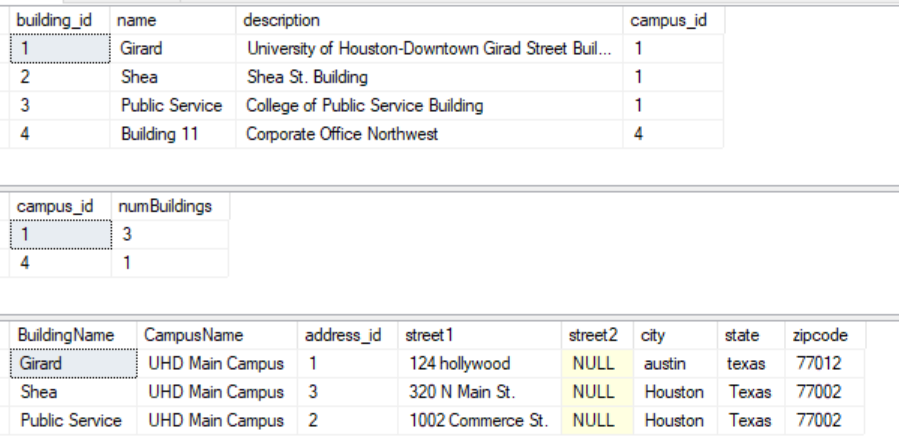
GROUP BY campus\_id

SELECT b.name BuildingName, c.description CampusName, a.\* FROM building b

join campus c on c.campus\_id = b.campus\_id

join Building\_Addr ba on ba.building\_id = b.building\_id

join Address a on a.address\_id = ba.address\_id



----------------------------------------

---- Building\_Addr

----------------------------------------

SELECT \* FROM building\_addr

--INSERT INTO Building\_Addr

--VALUES (4,4)

--UPDATE Building\_Addr

--SET address\_id = 4 WHERE building\_id = 4

--DELETE FROM Building\_Addr

--WHERE address\_id = 4 and building\_id = 4

SELECT building\_id, count(\*) addressCount

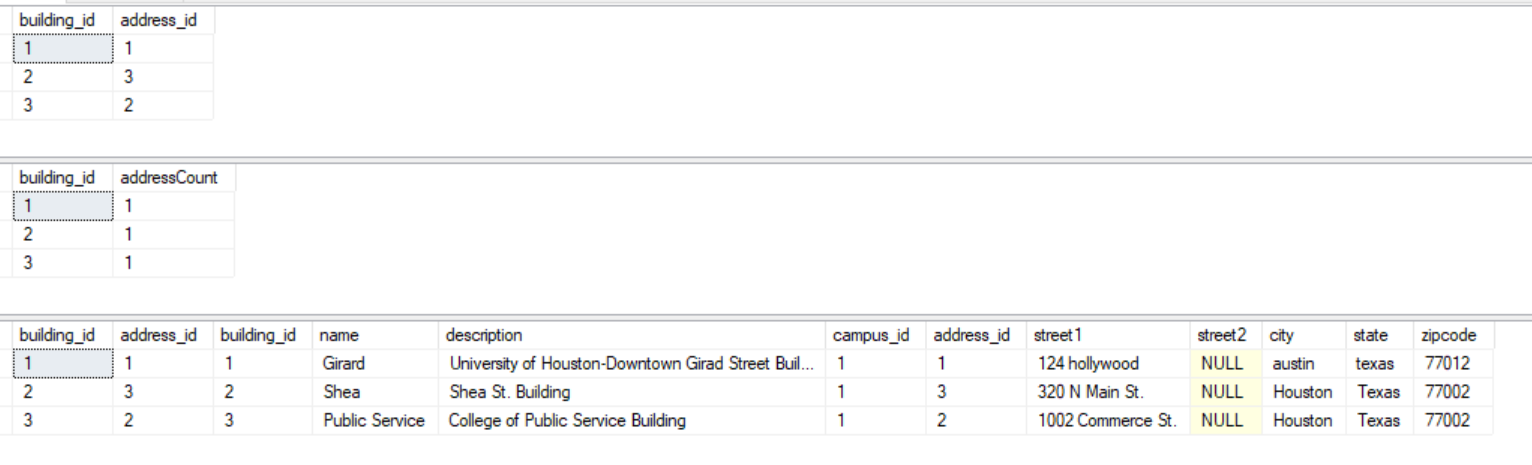
from Building\_Addr

Group by building\_id

SELECT \* FROM building\_addr ba

join building b on ba.building\_id = b.building\_id

join address a on a.address\_id = ba.address\_id



----------------------------------------

---- Room

----------------------------------------

SELECT \* FROM room

--INSERT INTO Room

--VALUES (12,1234,4,'Corporate Office Kingwood',30,1);

--UPDATE Room

--SET is\_active = 0

--WHERE room\_id = 11

--DELETE FROM Room

--WHERE room\_id = 12

SELECT r.building\_id, count(r.room\_id) num\_rooms

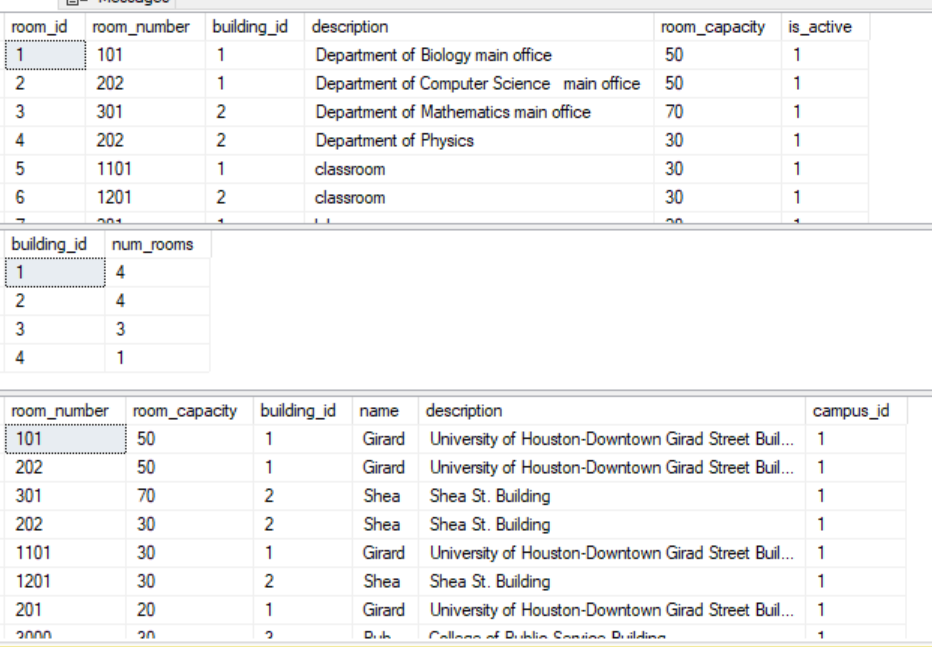
FROM room r

GROUP BY r.building\_id

SELECT r.room\_number, r.room\_capacity, b.\*

FROM room r

join building b on b.building\_id = r.building\_id



----------------------------------------

---- Student

----------------------------------------

SELECT \* FROM student --WHERE Student\_id = 7

--INSERT INTO Student (student\_id,fname,lname,dob,is\_active)

--VALUES (7,'Caitlin','Boake','1989-01-01',1);

--UPDATE Student

--SET DOB = '1989-12-15'

--WHERE student\_id = 7

--DELETE FROM Student

--WHERE student\_id = 7

SELECT count(is\_active) num\_active\_students

FROM Student

WHERE is\_active = 1

GROUP BY is\_active

SELECT \* FROM Student s

join Student\_Addr sa on sa.student\_id = s.student\_id

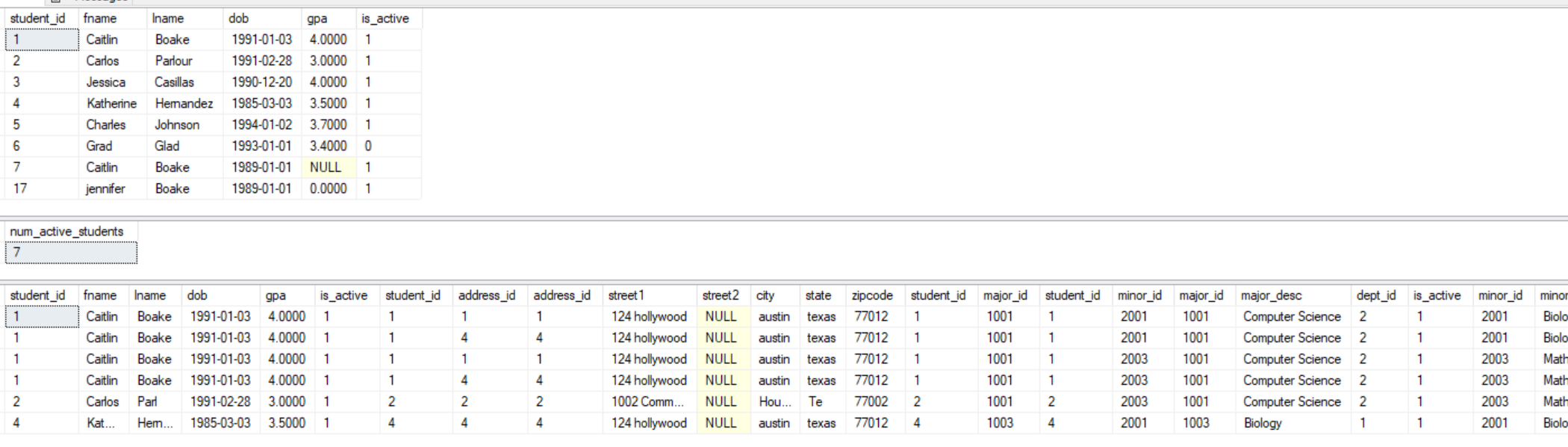
left outer join Address a on a.address\_id = sa.address\_id

left outer join Student\_Major smj on smj.student\_id = s.student\_id

left outer join Student\_Minor smn on smn.student\_id = s.student\_id

join Major m on m.major\_id = smj.major\_id

join Minor mn on mn.minor\_id = smn.minor\_id



----------------------------------------

---- Major

----------------------------------------

SELECT \* FROM major

--INSERT INTO Major

--VALUES (1005, 'English', 2,2)

--UPDATE major

--SET is\_active = 1, dept\_id = 5

--WHERE major\_id = 1005

--DELETE FROM Major

--WHERE major\_id = 1006

SELECT dept\_id, count(major\_id) num\_major

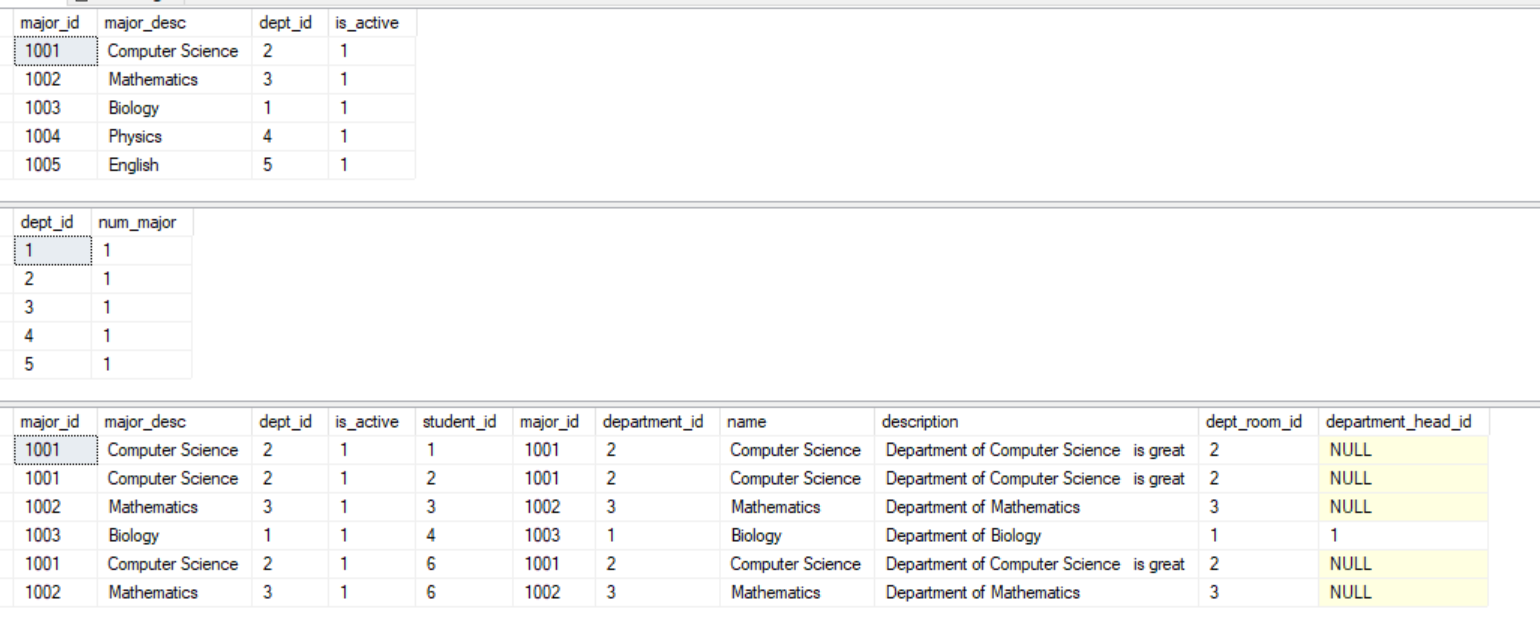
FROM major

GROUP BY dept\_id

SELECT \* FROM major m

join student\_major sm on sm.major\_id = m.major\_id

join Department d on d.department\_id = m.dept\_id



----------------------------------------

---- Minor

----------------------------------------

SELECT \* FROM minor

--INSERT INTO Minor

--VALUES (2005, 'English', 5,2)

--UPDATE Minor

--SET is\_active = 1, dept\_id = 5

--WHERE minor\_id = 2004

--DELETE FROM Minor

--WHERE minor\_id = 2005

SELECT dept\_id, count(minor\_id) num\_minor

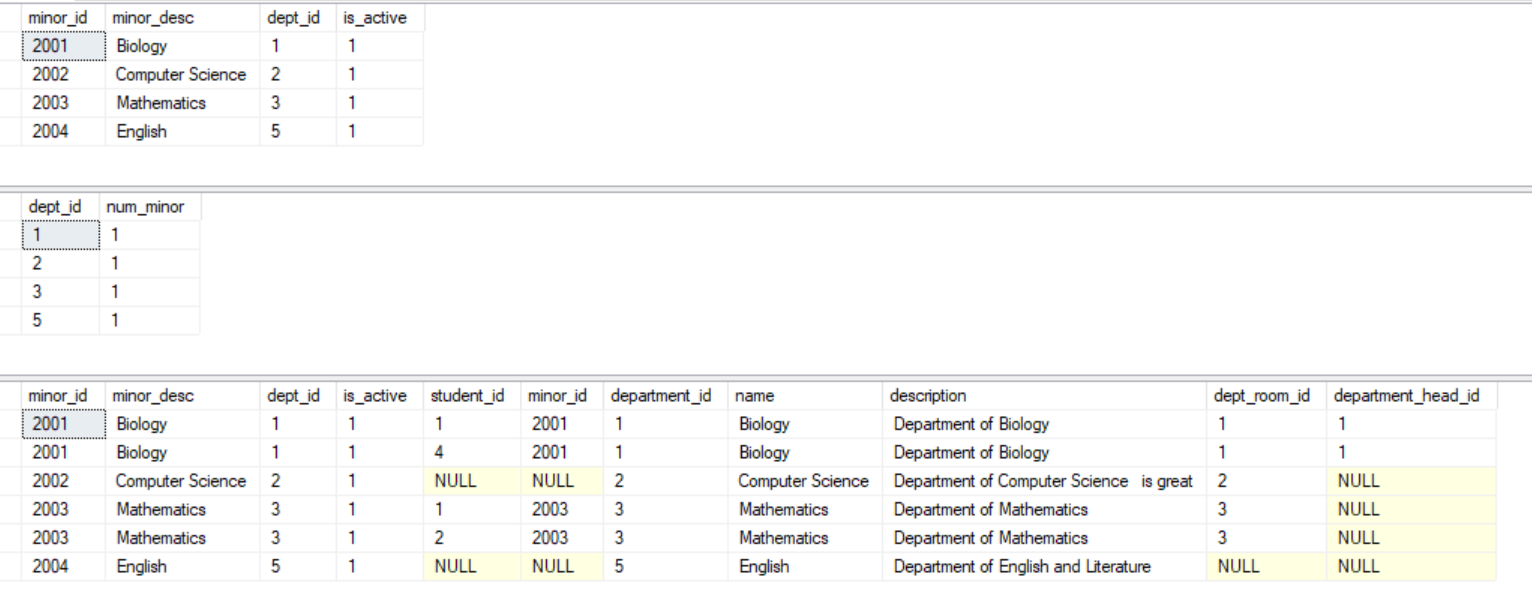
FROM minor

GROUP BY dept\_id

SELECT \* FROM Minor m

left outer join Student\_Minor sm on sm.minor\_id = m.minor\_id

join Department d on d.department\_id = m.dept\_id



----------------------------------------

---- Student\_Major

----------------------------------------

SELECT \* FROM student\_major WHERE Student\_id = 6

--INSERT INTO Student\_Major

--VALUES (6,1002)

--UPDATE Student\_Major

--SET major\_id = 1003

--WHERE student\_id = 6 and major\_id = 1002

--DELETE FROM Student\_Major

--WHERE student\_id = 6 and major\_id = 1003

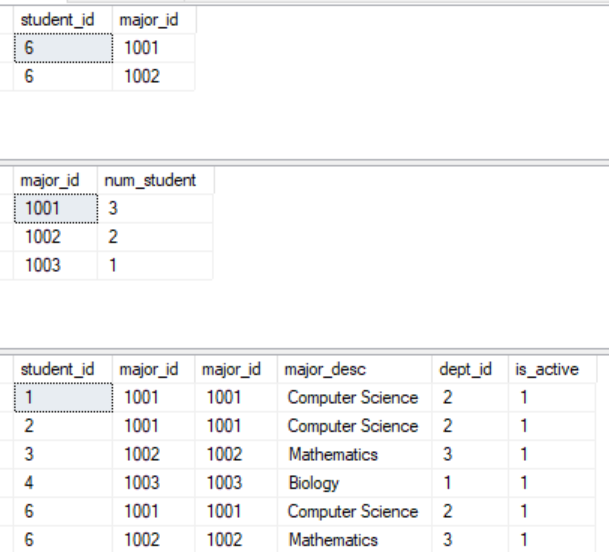
SELECT major\_id, count(distinct student\_id) num\_student

FROM student\_major

GROUP BY major\_id

SELECT \* FROM Student\_Major sm

join Major m on m.major\_id = sm.major\_id



----------------------------------------

---- Student\_Minor

----------------------------------------

SELECT \* FROM student\_minor

--INSERT INTO Student\_Minor

--VALUES (6,2004)

--UPDATE Student\_Minor

--SET minor\_id = 2002

--WHERE student\_id = 6 and minor\_id = 2004

--DELETE FROM Student\_Minor

--WHERE student\_id = 6 and minor\_id = 2002

SELECT minor\_id, count(student\_id) NumStudentsWithMinor

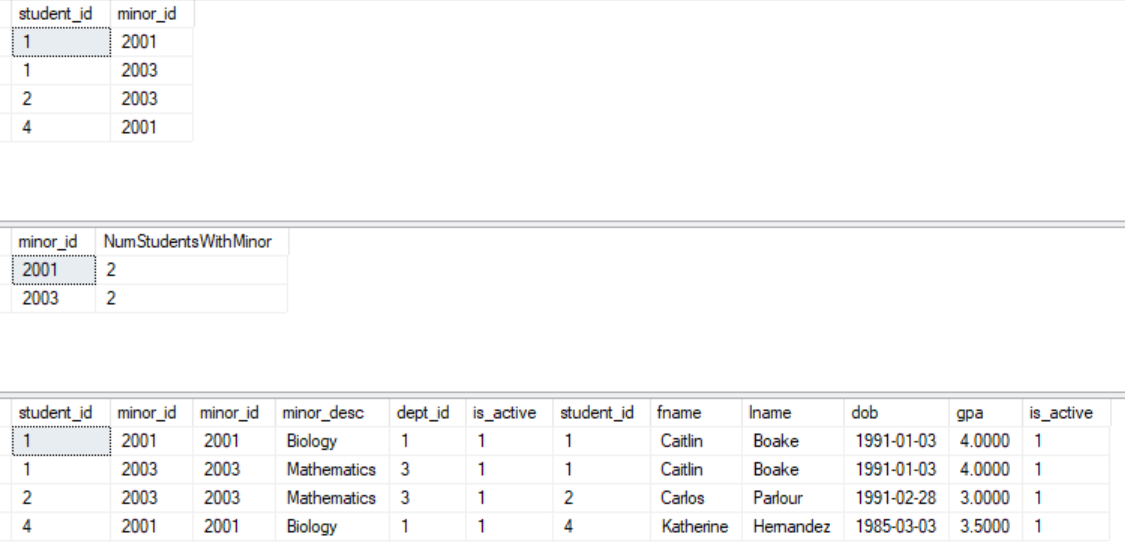
FROM student\_minor

GROUP BY minor\_id

SELECT \* FROM Student\_Minor sm

join Minor m on m.minor\_id = sm.minor\_id

join Student s on s.student\_id = sm.student\_id



------------------------------------------

---- Student\_Addr

----------------------------------------

SELECT \* FROM student\_addr

--INSERT INTO Student\_Addr

--VALUES (1,3)

--UPDATE Student\_Addr

--SET address\_id = 3

--WHERE student\_id = 1 and address\_id = 2

--DELETE FROM Student\_Addr

--WHERE student\_id = 1 and address\_id = 3

SELECT student\_id, count(address\_id) AddressesPerStudent

FROM student\_addr

GROUP BY student\_id

SELECT \* FROM Student\_Addr sa

join Address a on a.address\_id = sa.address\_id

join student s on s.student\_id = sa.student\_id



----------------------------------------

---- Staff

----------------------------------------

SELECT \* FROM staff

--INSERT INTO Staff

--VALUES (11,'Benjamin','Soibam','Dr.','2999-01-01','Assistant Professor','\\fileshare\c$\share\fileloc\filename2.pdf',1,2)

--UPDATE Staff

--SET cv\_file\_loc = '\\fileshare\c$\share\fileloc\filename3.pdf' WHERE staff\_id = 10

--DELETE FROM Staff

--WHERE staff\_id = 11

SELECT count(is\_active) num\_active\_staff

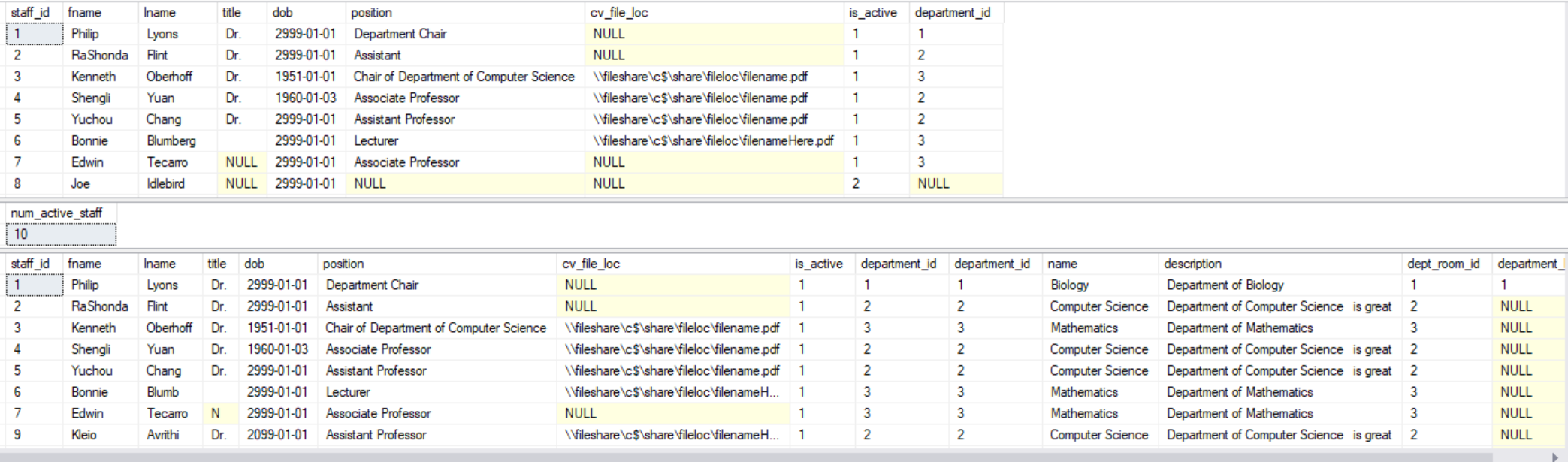
FROM Staff

WHERE is\_active = 1

GROUP BY is\_active

SELECT \* FROM Staff s

join Department d on d.department\_id = s.department\_id



----------------------------------------

---- Staff\_Addr

----------------------------------------

SELECT \* FROM staff\_addr

--INSERT INTO staff\_addr

--VALUES (2,5)

--UPDATE staff\_addr

--SET address\_id =2

--WHERE staff\_id = 2

--DELETE FROM staff\_addr

--WHERE staff\_id = 2

SELECT staff\_id, count(address\_id) AddressesPerStudent

FROM staff\_addr

GROUP BY staff\_id

SELECT \* FROM staff\_addr sa

join Address a on a.address\_id = sa.address\_id

join Staff s on s.staff\_id = sa.staff\_id



----------------------------------------

---- Department

----------------------------------------

SELECT \* FROM department

--INSERT INTO Department

--VALUES (6,'English','Department of English and Literature',null,null)

--UPDATE Department

--SET dept\_room\_id = 3

--WHERE department\_id = 6

--DELETE FROM Department

--WHERE department\_id = 6

SELECT COUNT(\*) numOfDepartments

FROM Department

SELECT \* FROM Department d

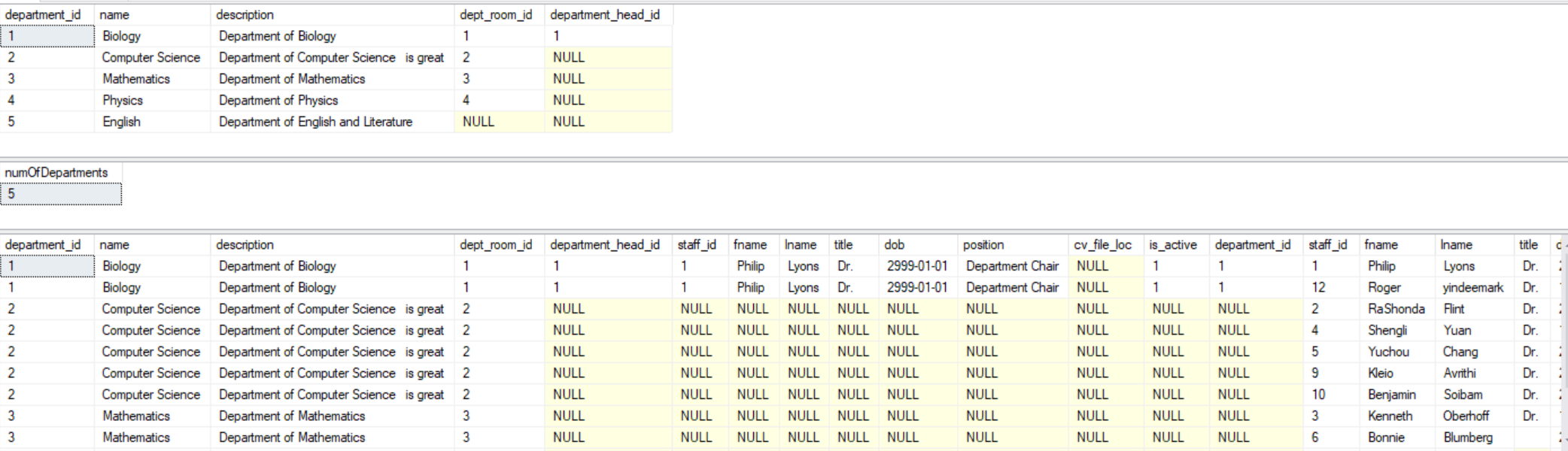
left outer join staff ds on ds.staff\_id = d.department\_head\_id

left outer join staff s on s.department\_id = d.department\_id

left outer join room r on r.room\_id = d.dept\_room\_id

left outer join Major mj on mj.dept\_id = d.department\_id

left outer join Minor mn on mn.dept\_id = d.department\_id



----------------------------------------

---- Course

----------------------------------------

SELECT \* FROM Course where course\_id = 40678--where course\_id = 40025

--INSERT INTO Course

--VALUES (40025,'Introduction to Python','Basic computer science principles using Python',0)

--UPDATE Course

--SET is\_active =1

--WHERE course\_id = 40023

--UPDATE Course

--SET is\_active = 2

--WHERE course\_id =40678

--DELETE FROM Course

--WHERE course\_id = 40025

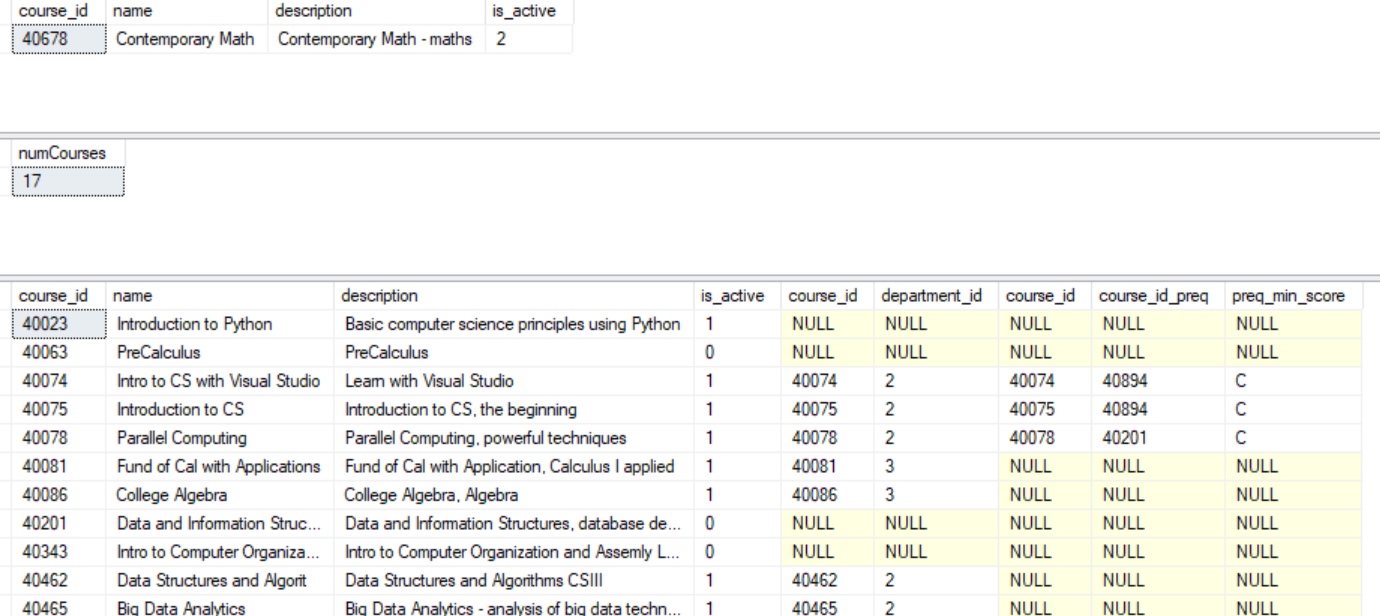
SELECT COUNT(\*) numCourses

FROM Course

SELECT \* FROM Course c

left outer join Course\_Dept cd on cd.course\_id = c.course\_id

left outer join Course\_Preq cpr on cpr.course\_id = c.course\_id



----------------------------------------

---- Course\_Dept

----------------------------------------

SELECT \* FROM course\_dept --where course\_id = 40608

--INSERT INTO Course\_Dept

--VALUES (40608,3)

--UPDATE Course\_Dept

--SET department\_id = 5

--WHERE course\_id = 40608 and department\_id = 3

--DELETE FROM Course\_Dept

--WHERE course\_id = 40608 and department\_id = 5

SELECT course\_id, COUNT(distinct department\_id) count\_dept

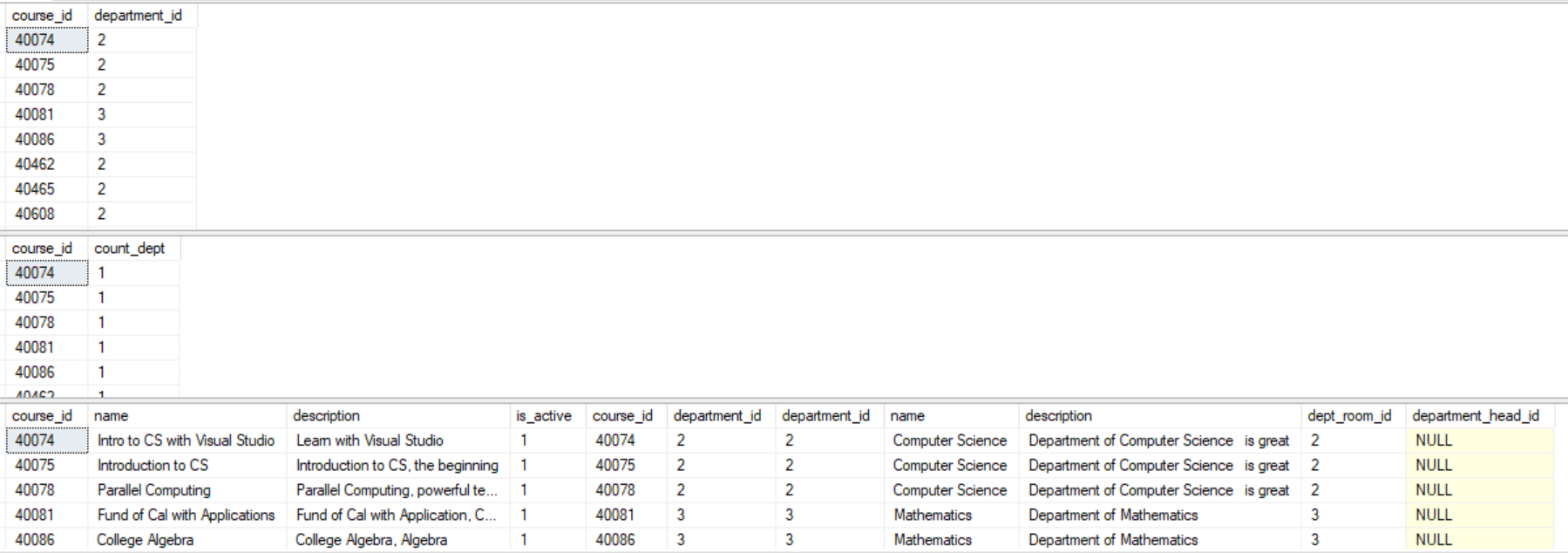
FROM course\_dept

GROUP BY course\_id

SELECT \* FROM Course c

join Course\_Dept cd on c.course\_id = cd.course\_id

join Department d on d.department\_id = cd.department\_id



----------------------------------------

---- Course\_Preq

----------------------------------------

SELECT \* FROM course\_preq

--INSERT INTO Course\_Preq

--VALUES (43290,40075,'B')

--UPDATE Course\_Preq

--SET preq\_min\_score = 'C'

--WHERE course\_id = 43290 and course\_id\_preq = 40075

--DELETE FROM Course\_Preq

--WHERE course\_id = 43290 and course\_id\_preq = 40081

SELECT course\_id, count(course\_id\_preq) num\_preq

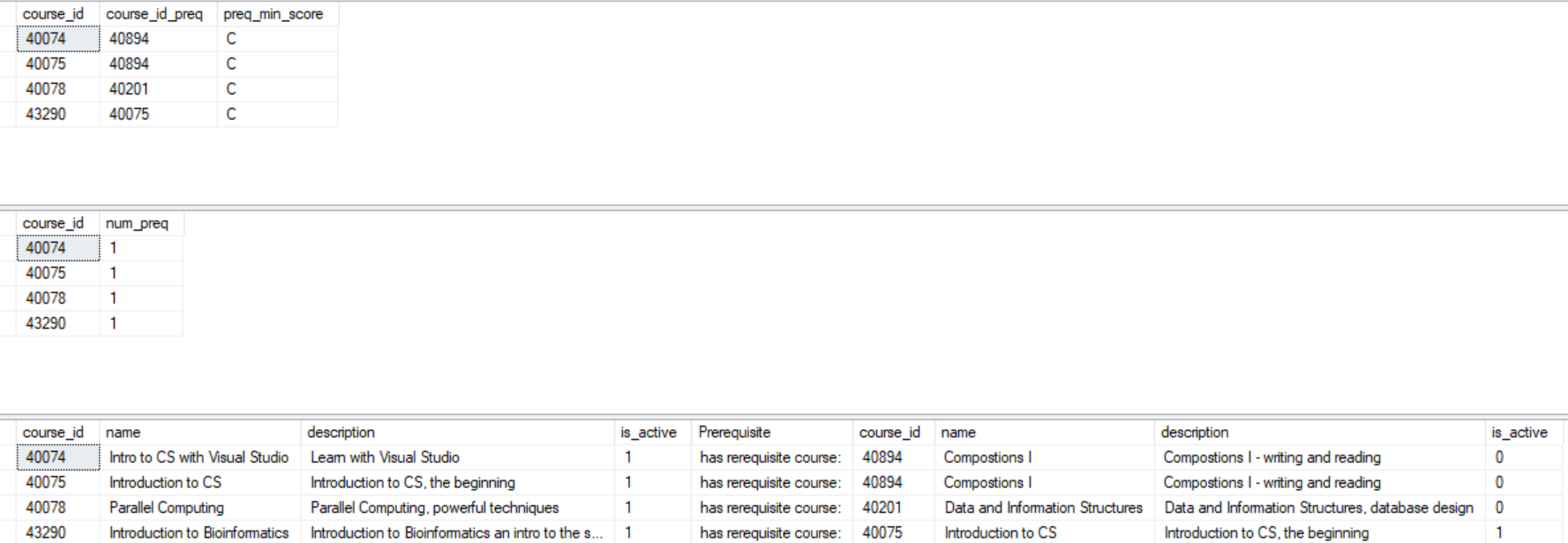
FROM course\_preq

GROUP BY course\_id

SELECT c.\*, 'has rerequisite course: ' Prerequisite, cpr.\* FROM course c

join course\_preq pr on pr.course\_id = c.course\_id

join course cpr on cpr.course\_id = pr.course\_id\_preq



----------------------------------------

---- Class\_Listing

----------------------------------------

SELECT \* FROM class\_listing where class\_listing\_id = 10

SELECT \* FROM Course

--INSERT INTO Class\_Listing

--VALUES (10,'Hybrid', 40023,2)

--UPDATE Class\_Listing

--SET professor\_id = 3

--WHERE class\_listing\_id = 8

--DELETE FROM Class\_Listing

--WHERE class\_listing\_id = 8

SELECT professor\_id, count(course\_id) num\_courses\_for\_professor

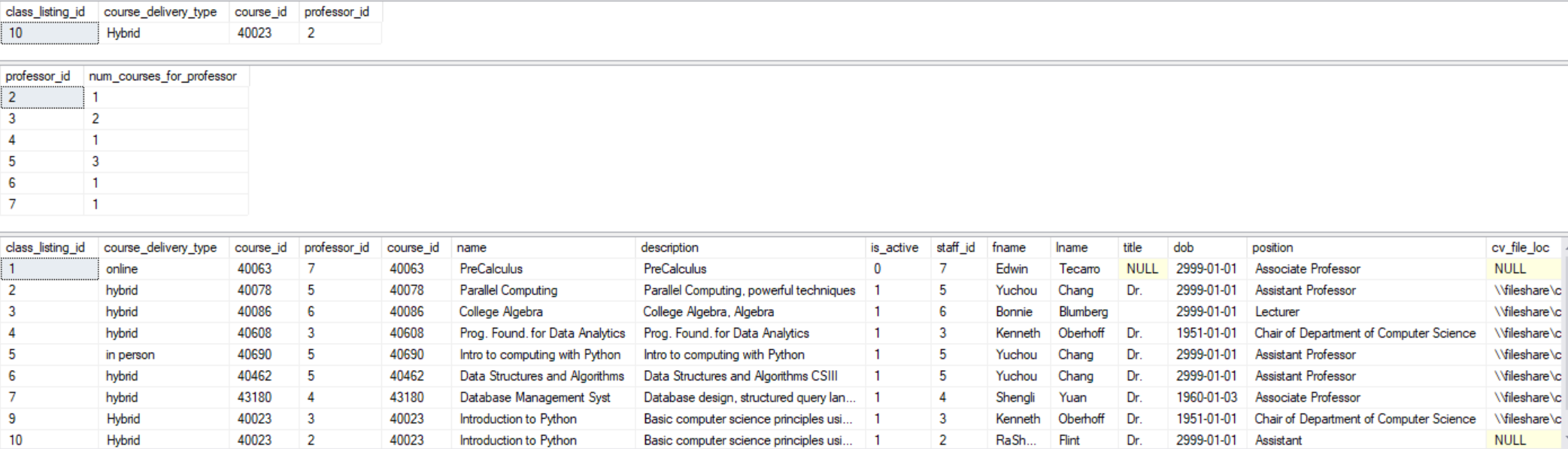
FROM class\_listing

GROUP BY professor\_id

SELECT \* FROM Class\_Listing cl

join Course c on c.course\_id = cl.course\_id

join staff s on s.staff\_id = cl.professor\_id



----------------------------------------

---- Class\_Room\_Occupancy

----------------------------------------

SELECT \* FROM class\_room\_occupancy where room\_occupancy\_id = 6

--INSERT INTO Class\_Room\_Occupancy

--VALUES (6,'TH','2018-07-09','2018-08-09','14:45','16:45',10,9)

--UPDATE Class\_Room\_Occupancy

--SET room\_id = 10 WHERE room\_occupancy\_id = 6

--DELETE FROM Class\_Room\_Occupancy

--WHERE room\_id = 10

SELECT room\_id, count(room\_id) room\_count

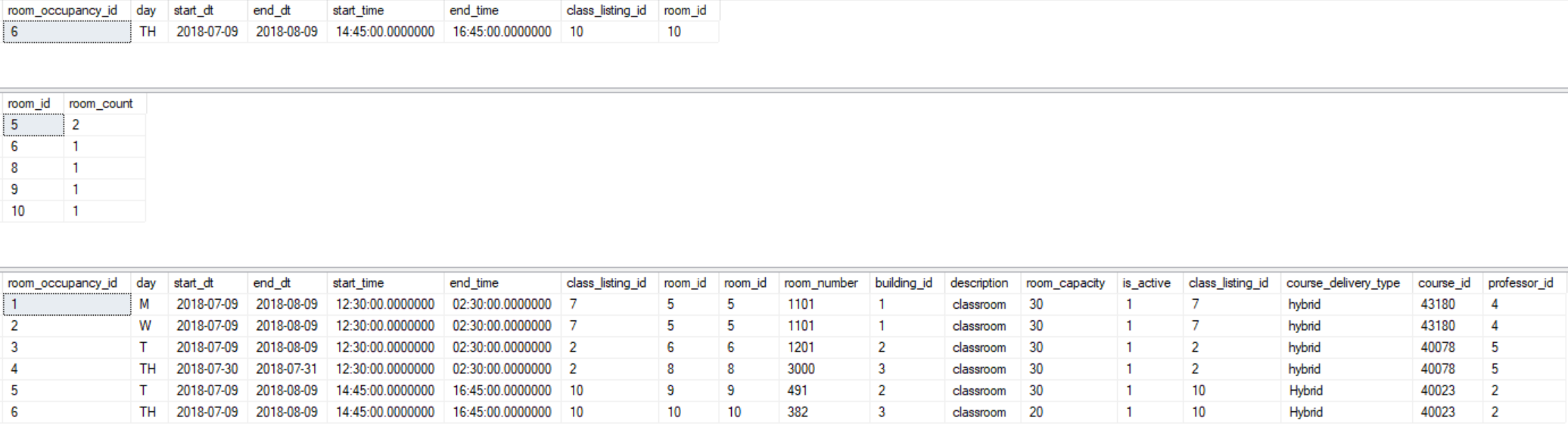
FROM class\_room\_occupancy

GROUP BY room\_id

SELECT \* FROM Class\_Room\_Occupancy cro

join Room r on r.room\_id = cro.room\_id

join Class\_Listing cl on cl.class\_listing\_id = cro.class\_listing\_id



----------------------------------------

---- Class\_Registration\_Listing

----------------------------------------

SELECT \* FROM class\_registration\_listing

--INSERT INTO Class\_Registration\_Listing (class\_listing\_id, student\_id)

--VALUES (4,4);

--UPDATE Class\_Registration\_Listing

--SET rating = 9, comment = 'Enjoyed the class, thank you.'

--WHERE student\_id = 1 and class\_listing\_id = 1

--UPDATE Class\_Registration\_Listing

--SET grade = 4.0

--WHERE class\_listing\_id = 7 and student\_id = 2

--DELETE FROM Class\_Registration\_Listing

--WHERE class\_listing\_id = 4 and student\_id = 4

SELECT class\_listing\_id, count(Student\_id) numStudents

FROM Class\_Registration\_Listing

group by class\_listing\_id

SELECT a.\*, crl.\*

FROM Class\_Registration\_Listing crl

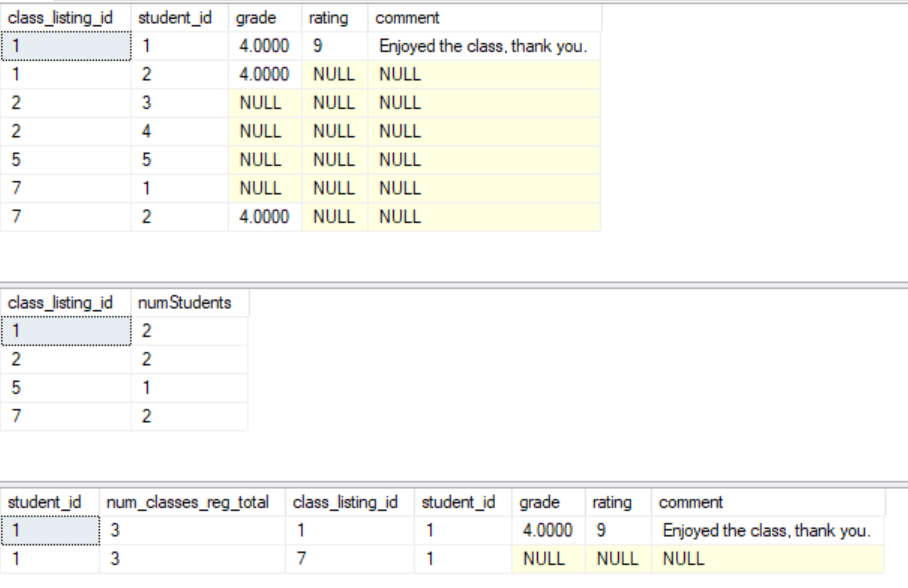
JOIN (

SELECT student\_id,count(class\_listing\_id) num\_classes\_reg\_total

FROM Class\_Reg\_hist\_Listing

WHERE student\_id = 1 and reg\_dt between '2018-07-01' and '2018-10-01' and dereg\_dt is null

GROUP BY student\_id) a on a.student\_id = crl.student\_id



----------------------------------------

---- Class\_Reg\_History\_Listing

----------------------------------------

SELECT \* FROM class\_reg\_hist\_listing

--INSERT INTO Class\_Reg\_Hist\_Listing(class\_listing\_id, student\_id, reg\_dt)

--SELECT class\_listing\_id, student\_id, CURRENT\_TIMESTAMP

--FROM Inserted

-- UPDATE crh

--SET dereg\_dt = CURRENT\_TIMESTAMP

--FROM Class\_Reg\_Hist\_Listing crh

--JOIN DELETED d on crh.class\_listing\_id = d.class\_listing\_id and crh.student\_id = d.student\_id

--WHERE crh.reg\_dt is not null

-- do not delete from this table

SELECT student\_id, count(\*) numClassesRegHist

FROM class\_reg\_hist\_listing

GROUP BY student\_id, class\_listing\_id

SELECT \*

FROM Student s

JOIN Class\_Reg\_Hist\_Listing crh on crh.student\_id = s.student\_id

join Class\_Listing cl on cl.class\_listing\_id = crh.class\_listing\_id

WHERE s.fname like '%Caitlin%'



----------------------------------------

---- Graduates

----------------------------------------

SELECT \* FROM graduates

--INSERT INTO Graduates

--VALUES(5,'2018-05-14',3.600)

--UPDATE Graduates

--SET final\_GPA = '3.5942'

--WHERE student\_id = 5

--DELETE FROM Graduates

--WHERE student\_id = 5

SELECT count(distinct student\_id) numberOfGrads

FROM graduates

SELECT \* FROM graduates g

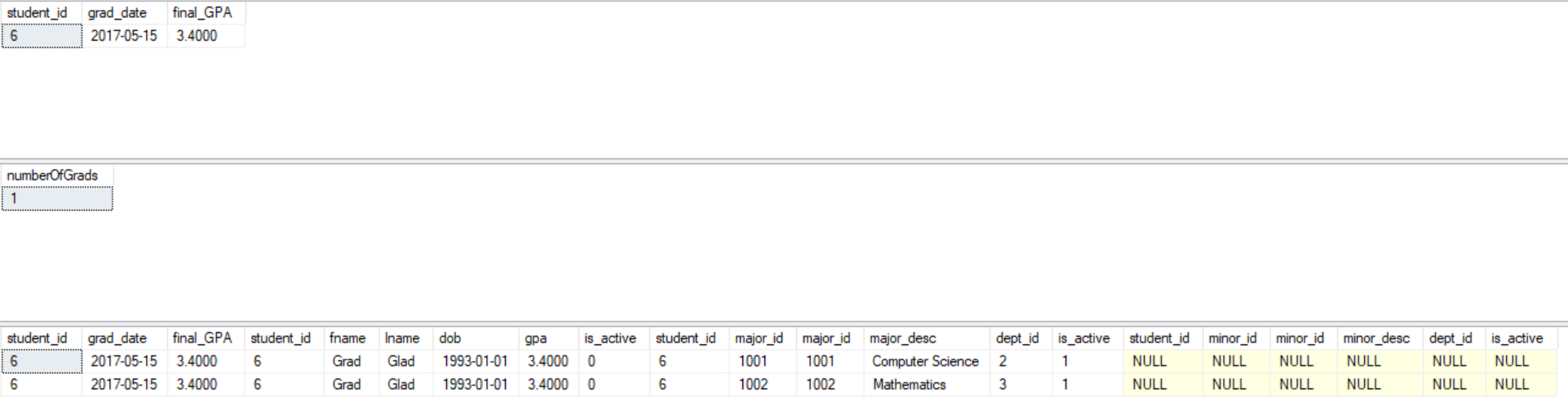
join student s on s.student\_id = g.student\_id

join Student\_Major smj on smj.student\_id = s.student\_id

join Major m on m.major\_id = smj.major\_id

left outer join Student\_Minor smn on smn.student\_id = s.student\_id

left outer join Minor mn on mn.minor\_id = smn.minor\_id



**Tables**

CREATE TABLE University (

University\_Id int not null,

Name varchar(255),

Description varchar(4000),

PRIMARY KEY(University\_Id)

);

CREATE TABLE Address (

address\_id int,

street1 varchar(255),

street2 varchar(255),

city varchar(255),

state varchar(255),

zipcode int,

PRIMARY KEY (address\_id)

);

CREATE TABLE Campus (

campus\_id int,

description varchar(512),

university\_id int REFERENCES University(university\_id),

PRIMARY KEY(campus\_id)

);

CREATE TABLE Campus\_Addr (

campus\_id int REFERENCES Campus(campus\_id) ON DELETE CASCADE,

address\_id int REFERENCES Address(address\_id) ON DELETE CASCADE,

PRIMARY KEY(campus\_id, address\_id)

);

CREATE TABLE Building (

building\_id int,

name varchar(255),

description varchar(512),

campus\_id int REFERENCES campus(campus\_id) ON DELETE SET NULL,

PRIMARY KEY(building\_id)

);

CREATE TABLE Building\_Addr (

building\_id int REFERENCES Building(building\_id) ON DELETE CASCADE,

address\_id int REFERENCES Address(address\_id) ON DELETE CASCADE,

PRIMARY KEY(building\_id, address\_id)

);

CREATE TABLE Room (

room\_id int,

room\_number varchar(255),

building\_id int REFERENCES Building(building\_id),

description varchar(512),

room\_capacity int,

is\_active tinyint,

PRIMARY KEY(room\_id)

);

CREATE TABLE Department (

department\_id int,

name varchar(255),

description varchar(512),

dept\_room\_id int REFERENCES Room(room\_id) ON DELETE SET NULL,

department\_head\_id int REFERENCES Staff(staff\_id) ON DELETE SET NULL,

PRIMARY KEY(department\_id)

);

CREATE TABLE Staff (

staff\_id int not null,

fname varchar(255) not null,

lname varchar(255) not null,

title varchar(5),

dob date,

position varchar(255),

cv\_file\_loc varchar(512),

is\_active tinyint,

PRIMARY KEY(staff\_id)

);

ALTER TABLE Staff ADD department\_id int REFERENCES Department(department\_id)

CREATE TABLE Staff\_Addr (

staff\_id int REFERENCES Student(student\_id) ON DELETE CASCADE,

address\_id int REFERENCES Address(address\_id) ON DELETE CASCADE,

PRIMARY KEY(staff\_id, address\_id)

);

CREATE TABLE Course (

course\_id int,

name varchar(255),

description varchar(4000),

is\_active tinyint,

PRIMARY KEY(course\_id)

);

CREATE TABLE Course\_Dept (

course\_id int REFERENCES Course(course\_id),

department\_id int REFERENCES Department(department\_id) ON DELETE SET NULL,

PRIMARY KEY(course\_id, department\_id)

);

CREATE TABLE Course\_Preq (

course\_id int REFERENCES Course(course\_id),

course\_id\_preq int REFERENCES Course(course\_id),

preq\_min\_score varchar(1),

PRIMARY KEY(course\_id, course\_id\_preq)

);

CREATE TABLE Class\_Listing (

class\_listing\_id int,

course\_delivery\_type varchar(255),

course\_id int REFERENCES Course(course\_id),

professor\_id int REFERENCES Staff(staff\_id),

PRIMARY KEY(class\_listing\_id)

);

CREATE TABLE Class\_Room\_Occupancy (

room\_occupancy\_id int,

day varchar(2),

start\_dt date,

end\_dt date,

start\_time time,

end\_time time,

class\_listing\_id int REFERENCES Class\_Listing(class\_listing\_id) ON DELETE CASCADE,

room\_id int REFERENCES Room(room\_id) ON DELETE CASCADE,

PRIMARY KEY(room\_occupancy\_id)

);

CREATE TABLE Student (

student\_id int,

fname varchar(255),

lname varchar(255),

dob date,

gpa decimal(5,4),

is\_active tinyint,

PRIMARY KEY(student\_id)

);

CREATE TABLE Graduates (

student\_id int REFERENCES Student(student\_id),

grad\_date date,

final\_GPA decimal(5,4),

PRIMARY KEY(student\_id, grad\_date)

);

CREATE TABLE Major (

major\_id int,

major\_desc varchar(255),

dept\_id int REFERENCES Department(department\_id) ON DELETE SET NULL,

is\_active tinyint,

PRIMARY KEY(major\_id)

);

CREATE TABLE Minor(

minor\_id int,

minor\_desc varchar(255),

dept\_id int REFERENCES Department(department\_id) ON DELETE SET NULL,

is\_active tinyint,

PRIMARY KEY(minor\_id)

);

CREATE TABLE Student\_Major(

student\_id int REFERENCES Student(student\_id) ON DELETE CASCADE,

major\_id int REFERENCES Major(major\_id) ON DELETE CASCADE,

PRIMARY KEY(student\_id, major\_id)

);

CREATE TABLE Student\_Minor(

student\_id int REFERENCES Student(student\_id) ON DELETE CASCADE,

minor\_id int REFERENCES Minor(minor\_id) ON DELETE CASCADE,

PRIMARY KEY(student\_id, minor\_id)

);

CREATE TABLE Student\_Addr (

student\_id int REFERENCES Student(student\_id) ON DELETE CASCADE,

address\_id int REFERENCES Address(address\_id) ON DELETE CASCADE,

PRIMARY KEY(student\_id, address\_id)

);

CREATE TABLE Class\_Registration\_Listing (

class\_listing\_id int REFERENCES Class\_Listing(class\_listing\_id),

student\_id int REFERENCES Student(student\_id),

grade decimal(5,4),

rating smallint,

comment varchar(4000),

PRIMARY KEY(class\_listing\_id, student\_id)

);

CREATE TABLE Class\_Reg\_Hist\_Listing (

class\_listing\_id int REFERENCES Class\_Listing(class\_listing\_id),

student\_id int REFERENCES Student(student\_id),

reg\_dt datetime,

dereg\_dt datetime,

PRIMARY KEY(class\_listing\_id, student\_id, reg\_dt)

);

**Database Prototype**

We are going to use SQL server to create our database.

**Project Time Table**

|  |  |  |
| --- | --- | --- |
| Task | Team Member | Date |
| Finish E/R Diagram |  |  |
| Complete Database Design |  |  |
| Complete Case Relations |  |  |
| Provide the complete SQL statements, or relational algebra operations for every use case. |  |  |
| Complete SQL Schema |  |  |
| Begin Constructing Database using SQL server |  |  |
| Test Database |  |  |
| Continue Database testing |  |  |
| Complete conclusion and reference |  |  |
| Complete Database Functionalities and User Interface Views |  |  |

**Conclusion**

**References**