4. Intermediate SQL

4.2.1 JOINs

Display the list of all departments, with the total number of instructors in each department. Note: Department(DeptName, Building, Budget) Instructor(InstID, InstName, DeptName, Salary)

Answer:
SELECT DeptName, COUNT(InstID)
FROM Department NATURAL LEFT OUTER JOIN Instructor GROUP BY DeptName;

DeptName	COUNT(InstID)
Biology	1
Comp. Sci.	3
Elec. Eng.	1
Finance	2
History	2
Music	1
Physics	2

LEFT OUTER is needed as departments might not have instructors.

4.2.2 JOINs

Display the list of active students, along with titles of the courses they take. The list should be sorted by the student names. Answer:
SELECT StudName, Title
FROM (Student NATURAL JOIN Takes) JOIN Course USING (CourseID) ORDER BY StudName;

StudName	Title
Aoi	Intro. to Digital Systems
Bourikas	Intro. to Computer Science
Bourikas	Robotics
Brandt	World History
Brown	Intro. to Computer Science
Brown	Image Processing
Chavez	Investment Banking
Levy	Intro. to Computer Science
Levy	Intro. to Computer Science
Levy	Image Processing
Peltier	Physical Principles
Sanchez	Music Video Production
Shankar	Intro. to Computer Science
Shankar	Game Design
Shankar	Robotics
Shankar	Database System Concepts
Tanaka	Intro. to Biology
Tanaka	Genetics
Williams	Intro. to Computer Science
Williams	Game Design
Zhang	Intro. to Computer Science
Zhang	Database System Concepts

Note, if LEFT OUTER was added to both JOINs then it would give an extra row (with 'NULL' for Title) for each student (like Snow) not taking any course.

Note, the following is a wrong answer, as it does not include students (like Levy and Bourikas in the relation instances in the slides) who take a course offered by a department different from the department to which the student belongs:
SELECT StudName, Title FROM (Student NATURAL JOIN Takes) NATURAL JOIN Course ORDER BY StudName; The reason for this is the fact that the second natural join requires Student.DeptName = Course.DeptName.

4.2.3 Referential Actions

Consider the CREATE TABLE commands for the university database in the appendix. Discuss why the referential ON DELETE actions have been chosen as they are.

Answer: ON DELETE CASCADE has been chosen for foreign keys in relations R, when the existence of tuples in R depend on the existence of a matching tuple in the referenced relation R'. With only one exception, this has been the case when the foreign key is part of the primary key of R. The exception is the PreReqID of the PreReq relation for which it has been chosen to have no referential action as it should not be allowed to remove a prerequisite of a course.

In the remaining cases (where a foreign key of a relation R is not part of the primary key R), ON DELETE SET NULL has been chosen, which allows the deletion of a referenced row by setting the referencing value in R to NULL. An alternative would have been to not have a referential action, disallowing such deletions, but in all the given cases this was found too restrictive. ON DELETE CASCADE has not been chosen as it deletes too much, e.g. if a department is deleted, we would not wish to delete associated instructors.

Consider the database instance in the appendix. Which table(s) are changed by the following command:

DELETE FROM COURSE **WHERE** CourseID = 'BIO-301';

Hint: First answer this without using your SQL DBMS.

Afterwards, you can check your answer by executing the command.

Answer: In the following tables, rows containing BIO-301 as CourseID, have been deleted: Course, PreReq, Section, Teaches, and Takes. Deletions in the last four tables are due to ON DELETE CASCADE actions. You can check this with the commands:

```
SELECT * FROM Course;
SELECT * FROM PreReq;
SELECT * FROM Section;
SELECT * FROM Teaches;
SELECT * FROM Takes;
```

4.2.4 Create a View

Called

SeniorInstructors(InstID, InstName, DeptName)

of instructors with a salary > 80000

Note:

Instructor(<u>InstID</u>, InstName, DeptName, Salary)

Answer:

```
CREATE VIEW SeniorInstructors AS
SELECT InstID, InstName, DeptName
FROM Instructor WHERE Salary > 80000;
```

4.2.5 Authorization

a) Connect to the database as Database Administrator (root) and create the users Karen, Linda and Susan with the generic password SetPassword. Please observe that user names are case sensitive!

```
CREATE USER 'Karen'@'localhost'
IDENTIFIED BY 'SetPassword';
CREATE USER 'Linda'@'localhost'
IDENTIFIED BY 'SetPassword';
CREATE USER 'Susan'@'localhost'
IDENTIFIED BY 'SetPassword';

SELECT user FROM mysql.user; -- shows users
```

b) Then grant SELECT to Karen and ALL to Linda and Susan to a database under your DBA control.

```
GRANT SELECT ON University.* TO
'Karen'@'localhost';
GRANT ALL ON University.* TO 'Linda'@'localhost';
GRANT ALL ON University.* TO 'Susan'@'localhost';

SHOW GRANTS FOR 'Karen'@'localhost';
SHOW GRANTS FOR 'Linda'@'localhost';
SHOW GRANTS FOR 'Susan'@'localhost';
```

c) Then close your connect to the server (on Workbench under Windows this can be done choosing File->Close Connection Tab).

d) Then add a connection for Karen. From the welcome page of Workbench this is done as follows:

1. Click on the + icon and a window will pop up.
2. In that window choose a name (e.g. KarenConnection) for the connection, set the username to Karen, check port is 3306, and click **Ok**. Then you will be returned to the welcome page where you can see the new connection KarenConnection.

e) Connect as Karen:

1. Click on the icon with the new connection name (KarenConnection) and a window will pop up.
2. Fill out the password choosen for Karen in step a): SetPassword. It will then probably give a connection warning and you should just click on **Continue Anyway.**

f) Change Karen's password to KarenSecret

```
SET PASSWORD =  PASSWORD('KarenSecret');
```

g) Try (still as Karen) to execute some select statements and some table modifications and see what happens.

SELECT statements should succeed for Karen, but modifications should be denied. Try e.g.

```
USE University;
SELECT * FROM Instructor;
DELETE FROM Instructor  WHERE DeptName =
'History';
```

h) Close the connection and then connect as DBA (root) and drop users Karen, Linda and Susan.

```
DROP USER 'Karen'@'localhost';
DROP USER 'Linda'@'localhost';
DROP USER 'Susan'@'localhost';

SELECT user FROM mysql.user;
```

i) Close the connection. Delete KarenConnection by clicking on the tools icon on the Workbench welcome page, then clicking on KarenConnection, and finally clicking on **delete** and then **close.**